ANALISIS PERBANDINGAN KONSUMSI DAYA *LIBRARY IMAGE LOADER* PADA *ANDROID*
(Studi Kasus: Aplikasi Media Sosial)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muhammad Abyan Safitra
NIM: 125150200111098

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

ANALISIS PERBANDINGAN KONSUMSI DAYA *LIBRARY IMAGE LOADER* PADA *ANDROID* (Studi Kasus: Aplikasi Media Sosial)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Abyan Safitra
NIM: 125150200111098

Skripsi ini telah diuji dan dinyatakan lulus pada
26 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Mahardeka Tri Ananta, S.Kom., M.T., M.Sc.	Komang Candra Brata, S.Kom., M.T., M.Sc.
NIK: 201607 8912041 001	NIK: 201607 8907111 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 2003121 001

IDENTITAS TIM PENGUJI

Penguji 1 :

Nama : Agi Putra Kharisma, S.T, M.T

NIP/NIK : 201304 860430 1 001

Penguji 2 :

Nama : Adam Hendra Brata, S.Kom., M.T., M.Sc.

NIP/NIK : 2016079001051001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 24 Juli 2018

Muhammad Abyan Safitra

NIM: 125150200111098

DAFTAR RIWAYAT HIDUP

Nama : Muhammad Abyan Safitra

Tempat, Tanggal Lahir : Malang, 4 Maret 1995

Riwayat Sekolah : MI Jenderal Sudirman

MTsN 1 Malang

SMK Telkom Sandhy Putra Malang

KATA PENGANTAR

Puji Syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena hanya dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan skripsi dengan judul "Analisis Perbandingan Konsumsi Daya *Library Image loader* pada *Android* (Studi Kasus: Aplikasi Media Sosial)". Melalui kesempatan ini, penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan selama penulisan skripsi ini, diantaranya:

1. Bapak Mahardeka Tri Ananta, S.Kom., M.T., M.Sc., selaku Dosen Pembimbing pertama yang telah membimbing dan telah memberikan ilmu serta saran dalam menyelesaikan skripsi ini.
2. Bapak Komang Candra Brata, S.Kom., M.T., M.Sc.selaku Dosen Pembimbing kedua yang telah membimbing dan telah memberikan ilmu serta saran dalam menyelesaikan skripsi ini.
3. Kedua Orang Tua Penulis serta keluarga besar atas segala doa, nasehat, dukungan baik moril maupun materiil dalam melancarkan skripsi ini.
4. Staff beserta dosen Fakultas Ilmu Komputer Universitas Brawijaya yang telah membekali penulis berbagai ilmu selama mengikuti perkuliahan sampai akhir penulisan skripsi.
5. Sahabatku Harisuddin dan Ferryansyah serta kawan-kawan mahasiswa FILKOM UB angkatan 2011 dan 2012 atas bantuan, dukungan, motivasi dan berbagi informasi yang turut membantu peyelesain skripsi ini.
6. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat secara langsung atau tidak langsung agar terselesaikannya skripsi ini.

Dengan segala kerendahan hati, penulis menyadari bahwa skripsi ini masih memiliki banyak kekurangan. Oleh karena itu kritik dan saran yang bersifat konstruktif sangat dibutuhkan sebagai pedoman untuk menyempurnakan skripsi ini agar lebih baik. Penulis berharap semoga skripsi ini dapat bermanfaat bagi diri sendiri maupun bagi semua pihak.

Malang, 24 Juli 2018

Penulis

abyansafitra@gmail.com

ABSTRAK

Kemampuan daya baterai *smartphone* tidak bisa mengimbangi kemajuan teknologi lainnya dalam meningkatkan performa seperti *CPU* dan *memory*, sehingga perlu dilakukan penelitian agar dapat mengembangkan aplikasi *mobile* yang memiliki tingkat konsumsi daya yang rendah. Menurut survey, aplikasi media sosial adalah aplikasi yang paling sering digunakan oleh pengguna *smartphone*. Salah satu cara agar dapat mengembangkan aplikasi *mobile* yang bisa memliki tingkat konsumsi daya yang rendah adalah dengan mengoptimalkan penggunaan *library*. *Library Image loader* adalah salah satu *library* pada *android* yang banyak digunakan dalam aplikasi media sosial untuk menangani transformasi gambar. *Library Image loader* yang akan diuji pada penelitian ini adalah *library Picasso, Universal image loader (UIL)* dan *Glide*. Penelitian dilakukan dengan cara membuat purwarupa aplikasi media sosial yang menyerupai linimasa aplikasi *instagram* yang akan digunakan sebagai sistem *dummy*. Pengujian akan dilakukan selama rentang waktu tertentu dan dengan menggunakan beberapa parameter ukuran gambar yang berbeda untuk melihat pengaruh perbedaan ukuran gambar terhadap tingkat konsumsi daya *Library Image loader*. Hasil penelitian menunjukkan bahwa *library Glide* yang punya hasil rata-rata konsumsi daya yang paling rendah yaitu sebesar 225,780 mW. Sementara untuk pengujian dengan ukuran gambar yang berbeda, hanya berpengaruh pada *library Picasso* dan *UIL,* yaitu semakin besar ukuran gambar, maka semakin besar pula rata-rata tingkat konsumsi daya yang dibutuhkan.

Kata kunci: *Android*, Media Sosial, Konsumsi Daya, *Image loader* , *Trepn profiler*

ABSTRACT

The battery power of the smartphone can not keep pace with other technological advances in improving performance such as CPU and memory, so research needs to be done in order to develop mobile apps that have low power consumption levels. According the survey, from the many existing mobile apps, media sosial apps are the most used apps by smartphone users. One way to develop mobile applications that can have low power consumption is to optimize library usage. Library Image loader is one of the most popular android libraries used in media sosial applications to handle image transformation. Library Image loader which will be tested in this research is library Picasso, Universal image loader (UIL) and Glide. The research is done by making a media sosial prototype application that resembles instagram timeline that will be used as a dummy sistem. The test will be performed over a specified time range and by using several different image size parameters to see the effect of image size differences on the power level of the Library Image loader. The results showed that the Glide library had the lowest average power consumption of 225,780 mW. While for testing with different image sizes, it only affects the Picasso and UIL libraries, that is the larger the image size, the greater the average power consumption level required.

Keywords: Android, Media sosial, Power Consumption, *Image loader*, *Trepn profiler*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Keterbatasan energi baterai *smartphone* telah menjadi masalah di seluruh dunia. Pada *smartphone*, kemampuan daya baterai tidak bisa mengimbangi kemajuan teknologi lainnya dalam meningkatkan performa (seperti *CPU* dan *memory*). Hal ini sangat disayangkan mengingat kebutuhan konsumsi daya *smartphone* untuk konten multimedia cukup besar (Trestian, 2012).

Ada beberapa parameter yang bisa digunakan dalam pengukuran performa aplikasi pada *smartphone*, diantaranya *response time*, *CPU usage*, *memory usage*, *disk space* serta *battery usage* (Willockx, M., et al , 2016). Dari beberapa parameter tersebut, ada salah satu parameter penting yang biasanya menjadi pertimbangan penting bagi *user*. Studi menunjukkan bahwa konsumsi baterai adalah prioritas utama bagi pembeli *smartphone*. Menurut survei *IDC (International Data Corporation)* menunjukkan bahwa 56% pembeli *Android*, 49% pembeli *iPhone*, dan 53% pembeli *Windows Phone* mengatakan bahwa konsumsi baterai adalah alasan utama ketika mereka membeli *smartphone* (Medium, 2017). Sehingga konsumsi daya akan dijadikan sebagai parameter utama dalam penelitian ini. Pada sisi yang lain, *smartphone* perlu terhubung dengan internet agar dapat mengeluarkan performa yang maksimal dan dapat mengakomodir kebutuhan konsumen.

Hasil dari survey perilaku pengguna internet di Indonesia berdasarkan konten internet yang diakses menunjukkan bahwa ada 6 konten yang paling sering diakses yaitu: media sosial (97,4%), hiburan (96,8%), berita (96,4%), pendidikan (93,8%), komersil (93,1%), dan terakhir layanan public (91,6%) (APJII, 2016). Sementara Kominfo mengungkapkan pengguna internet di Indonesia saat ini mencapai 63 juta orang. Dari angka tersebut, 95 persennya menggunakan internet untuk mengakses media sosial (Kominfo, 2013). Saat ini teknologi internet dan *smartphone* makin maju, maka media sosial pun ikut tumbuh dengan pesat, bahkan sekarang terjadi fenomena baru pada penyebaran informasi yaitu peran media sosial yang sudah mulai tampak menggantikan peranan media masa konvensional dalam menyebarkan berita-berita (Azisubekti, 2016). Sehingga pada penelitian ini, media sosial akan digunakan sebagai fokus utama dalam studi kasus penelitian.

Dalam pengembangan aplikasi *smartphone*, *developer* bisa menggunakan beberapa *library*. Tujuan penggunaan *library* ialah untuk mempermudah proses pembuatan aplikasi. Sebuah *library* akan berguna pada saat *developer* sedang membangun beberapa aplikasi yang menggunakan beberapa komponen yang sama, seperti *Activity, service*, atau *layout user interface* (*Android Developer*, 2017).

Salah satu *library* pada *android* yang digunakan dalam aplikasi media sosial adalah *library image loader*. Keuntungan menggunakan *library image loader* adalah *developer* bisa lebih efisien jika ingin menampilkan beberapa gambar

karena *library* ini bisa menangani masalah *memory* dan *disk caching*. Selain itu *library* ini mempunyai beberapa fungsi untuk transformasi gambar seperti mengubah ukuran atau memotong gambar (Liutova, 2016). *Library image loader* yang banyak digunakan dari keseluruhan aplikasi yang ada pada *Google Playstore* yaitu *Picasso* dengan persentase 6,84%, kemudian *Universal image loader (UIL)* sebanyak 5,00%, dan *Glide* sebanyak 2,40% (AppBrain, 2017). Sehingga pada penelitian ini *library Image loader* yang akan dibandingkan adalah *Picasso, UIL* dan *Glide*.

Ada tiga metode yang dapat digunakan dalam pengukuran konsumsi daya pada perangkat *android*. Ketiga metode tersebut adalah: pertama, pengukuran dengan menggunakan *hardware* tambahan sebagai alat pengukur, kedua, menggunakan aplikasi *android* tambahan sebagai media pengukuran, dan yang ketiga adalah pengukuran menggunakan aplikasi pengukuran yang dipasang pada sebuah komputer dan terhubung pada *smartphone android* (Yang, P., Zhou, D., 2015). Masih menurut Yang, pengukuran konsumsi daya dengan bantuan aplikasi *android* tambahan untuk mengukur konsumsi daya merupakan metode yang paling banyak digunakan, dan memiliki pendekatan yang cukup baik untuk mengetahui tingkat konsumsi daya. Sehingga dalam penelitian mengenai perbandingan konsumsi daya *library image loader* ini akan dibuat sebuah purwaruapa aplikasi media sosial yang menyerupai timline aplikasi *instagram* yang akan digunakan sebagai sistem *dummy* untuk menguji konsumsi daya *library image loader* serta pengaruh ukuran gambar terkait dengan tingkat konsumsi dayanya.

Ada beberapa aplikasi yang cukup akurat untuk melakukan pengukuran konsumsi daya diantaranya yaitu: *Powertutor* dan *Trepn profiler* (Bakker, 2015). Namun untuk *powertutor*, sekarang ini masih belum ada pengembangan terbaru, sehingga sering tidak kompatibel dengan *smartphone android* dan sistem operasi *android* keluaran terbaru. Semetara untuk *Trepn profiler* masih terus dikembangkan sehingga masih bisa mengikuti perkembangan *chipset* terbaru dan sistem operasi *android* terbaru. Sehingga pada penelitian ini akan menggunakan aplikasi *Trepn profiler* untuk mengukur tingkat konsumsi daya per-aplikasi pada *android*.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah terurai di atas, peneliti merumuskan beberapa permasalahan yang akan dibahas pada penelitian ini, yaitu:

1. Bagaimana cara membandingkan konsumsi daya dari setiap *library picasso, glide* dan *UIL* ?
2. Bagaimana hasil perbandingan konsumsi daya *pada library picasso, glide* dan *UIL* pada studi kasus aplikasi media sosial berbasis *android*?
3. Bagaimana pengaruh perbedaan ukuran gambar terhadap *library picasso, glide* dan *UIL* yang diuji?

1.3 Tujuan

Penelitian yang dilakukan memiliki tujuan sebagai berikut:

1. Mengetahui cara menguji setiap *library* sehingga mendapatkan nilai konsumsi daya yang digunakan oleh setiap *library*.
2. Mengetahui hasil analisis perbandingan konsumsi daya dari setiap *library* yaitu *Picasso, glide dan UIL* pada *platform android* khususnya pada studi kasus aplikasi media sosial.
3. Mengetahui pengaruh perbedaan ukuran gambar yang berbeda terhadap kosumsi daya dari hasil pengujian setiap *library* yaitu *Picasso, glide dan UIL*.

1.4 Manfaat

Manfaat yang diharapkan dalam penelitian ini adalah untuk memberikan informasi bagi *developer* aplikasi media sosial dalam menentukan *library image loader* mana yang sesuai dengan kebutuhan khususnya pada platform *android*. Serta menjadi acuan bagi *developer* dalam mengembangkan aplikasi berbasis *android*.

1.5 Batasan masalah

1. Studi kasus yang digunakan ialah sebuah sistem *dummy* berupa purwarupa aplikasi media sosial yang menggunakan *library image loader* untuk penanganan gambarnya.
2. *Platform* yang digunakan pada penelitian ini dibatasi pada *platform android.*
3. Pengujian penggunaan daya dilakukan dengan bantuan aplikasi pengukur konsumsi daya untuk memantau tingkat konsumsi daya pada *android*.
4. Parameter pembanding dalam pengujian konsumsi daya *library image loader android* dibatasi pada parameter *memory* dan *CPU.*
5. *SDK (Software Development Kit) Android* minimal yang digunakan pada penelitian ini minimal *SDK* 16 (*jelly bean*).

1.6 Sistematika pembahasan

Sistematika dalam penulisan skripsi pada penelitian ini sebagai berikut:

BAB I : Pendahuluan

Bab ini menguraikan tentang latar belakang, rumusan masalah, tujuan, manfaat, dan batasan penelitian dalam Analisis Perbandingan Konsumsi Daya *Library Image loader* pada *Android*.

BAB II : Tinjauan Pustaka

Bab ini menguraikan mengenai kajian pustaka dari penelitian serupa yang pernah dilakukan sebelumnya dan teori-teori dalam penelitian Analisis Perbandingan Konsumsi Daya *Library Image loade r*pada *Android*.

BAB III : Metodologi

Bab ini menguraikan tentang langkah kerja dan metode yang diterapkan dalam Analisis Perbandingan Konsumsi Daya *Library Image loader* pada *Android*.

BAB IV : Analisis Kebutuhan dan Perancangan

Bab ini menguraikan tentang kebutuhan sistem dalam Analisis Perbandingan Konsumsi Daya *Library Image loader* pada *Android*.

BAB V : Implementasi

Bab ini akan menguraikan tahapan-tahapan dalam perancangan dan implementasi dalam Analisis Perbandingan Konsumsi Daya *Library Image loader* pada *Android*.

BAB VI : Pengujian dan Analisis

Bab ini menjelaskan tahapan pengujian dari analisis dari hasil yang telah didapat dalam pengujian

BAB VII: Penutup

Bab ini berisi kesimpulan dari penelitian yang telah dilakukan. Serta saran yang dapat digunakan untuk pengembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Bab Landasan Kepustakaan berisi teori-teori dasar dan penelitian sebelumnya untuk menunjang penelitian analisis konsumsi daya pada *image loader library android*.

2.1 Kajian Pustaka

Tabel 2.1 Kajian pustaka penelitian sebelumnya

Judul Penelitian	Peneliti	Penelitian yang dilakukan
Comparing Energy Profilers for Android	Bakker , A (2014)	Penelitian ini membandingan beberapa aplikasi pengukur konsumsi daya pada *smartphone android*. Ada 6 aplikasi pengukur konsumsi daya yang dibandingkan pada penelitian ini, aplikasi tersebut dibandingkan berdasarkan fungsionalitas dalam pengukuran. Salah satu parameter yang cukup penting adalah pengukuran konsumsi daya per-aplikasi yang sedang berjalan. Hasil dari penelitian ini ada 3 aplikasi yang cukup baik dalam mengukur konsumsi daya, yaitu *PowerTutor, Trepn profiler* dan *Intel Performance Viewer.*
A Light Weight Energy Profiling Approach on Android Devices	Yang, P., Zhou, D. (2015)	Jurnal ini menganalisis kelebihan dan kekurangan tiga metode dalam pengukuran konsumsi daya pada perangkat *android*. Ketiga metode tersebut adalah: pertama pengukuran dengan tambahan hardware tambahan sebagai alat pengukur, kedua menggunakan aplikasi *android* tambahan sebagai media pengukuran, dan yang ketiga adalah pengukuran menggunakan aplikasi pengukuran yang dipasang pada sebuah komputer dan terhubung pada *smartphone android*. Hasil dari penelitian menyarankan menggunakan pengukuran kedua, yaitu dengan bantuan aplikasi *android* tambahan

Tabel 2.1 Kajian pustaka penelitian sebelumnya (Lanjutan)

Judul Penelitian	Peneliti	Penelitian yang dilakukan
		untuk mengukur konsumsi daya, karena merupakan metode yang paling banyak digunakan, dan juga pendekatan yang cukup baik untuk mengetahui tingkat konsumsi daya. Sementara aplikasi yang disarankan adalah aplikasi *Trepn profiler*.
Optimizing Energy Consumption per Applications on Mobile Devices	Carlos, J. et al (2013)	Penelitian ini menganalisis konsumsi baterai dari setiap aplikasi yang berjalan pada *Android*, Aplikasi yang digunakan untuk meninjau analisis adalah *Powertutor,* yaitu aplikasi yang memberi Informasi tentang komponen dari ponsel yang mengkonsumsi daya tertinggi, dan kemudian hasilnya digunakan sebagai acuan untuk analisis.
A Study on Comparison Analysis of Performance and Usage between Picasso and Glide	Song Y, et al (2015)	Penelitian membandingkan kelebihan dan kekurangan dari *library Picasso* dan *Glide*, utamanya pada aspek penggunaan *memory*.

2.2 Gambar Digital

Sebuah gambar digital adalah representasi numerik dari (biasanya biner) gambar dua dimensi. Tergantung pada apakah resolusi gambar adalah tetap, biasanya istilah gambar digital mengacu pada gambar *bitmap*. Gambar *bitmap* adalah sebuah struktur data yang mewakili susunan piksel warna yang ditampilkan pada layar monitor, kertas atau media tampilan lainnya. Secara teknis gambar *bitmap* digambarkan dengan lebar dan tinggi dalam piksel dan dalam angka bit per piksel. Beberapa format gambar *bitmap* yang sering dijumpai: *GIF, JPEG, BMP dan PNG* (Raster Scracth-Wiki, 2017).

2.3 *Library Image loader Android*

Library Image loader adalah *library* pada *android* yang biasa digunakan untuk menangani berbagai permasalahan serta menangani berbagai kebutuhan terkait penggunaan gambar pada *android*. Berbagai hal yang bisa ditangani *library* ini

antara lain mengunduh gambar, modifikasi gambar serta mengoptimalkan penggunaan *memory* agar tidak menggunakan *resource* yang besar.

Beberapa projek aplikasi dapat dibuat dengan referensi *library* yang sama. Ada perbedaan yang signifikan dari aplikasi *android* dan *library android*. *Library* tak bisa di*compile* menjadi file *APK*, sebaliknya *library* menyediakan kode yang dapat digunakan kembali/*reuseable* (sitepoint, 2013). Pada penelitian kali ini, *library* yang digunakan adalah *library image loader*, yaitu *library* yang menangani segala hal tentang pemuatan gambar pada *android*. Ada 3 buah *library image loader* yang digunakan pada penelitian ini, yaitu: *Universal image loader (UIL), Glide dan Picasso.*

2.3.1 *Universal image loader* (*UIL*)

Sebagai *library* no 1 di *GitHub*, *UIL* bertujuan untuk memberikan alat yang kuat, fleksibel dan sangat disesuaikan untuk *loading*, *caching* dan menampilkan gambar. Ini memberikan banyak pilihan konfigurasi dan kontrol yang baik atas pemuatan gambar dan proses *caching* (*UIL Github, 2011*). *UIL* adalah sebuah *library* yang cerdas dan kuat yang membantu dalam pemuatan, *caching* dan menampilkan gambar pada *Android*. Ini berarti, menggunakan *library* ini Anda dapat men-download gambar dari *server* dan ditampilkan pada *ImageView*.

Fitur dari *UIL* adalah:

1. *Asynchronous* dan *multi-threading* loading gambar. Hal ini memungkinkan Anda untuk men-download beberapa gambar secara tidak berbarengan.
2. Mendukung berbagai konfigurasi yang membantu untuk memenuhi kebutuhan *developer*. Dengan ini *developer* dapat mengontrol *memory*, jenis *cache*, *decoder*, opsi tampilan gambar, dll.
3. Memungkinan *caching* gambar dalam *memory* dan/atau sistem file (atau kartu SD) perangkat.
4. Memungkinan untuk menampilkan proses *loading* pada saat mengunduh. Memungkinkan berbagai metode *callback* menggunakan yang Anda akan mendapatkan untuk mengetahui perkembangan/keadaan permintaan download (Stacktips, 2014).

Halaman mengenai *library UIL* dapat diakses pada:
https://github.com/nostra13/Android-Universal-Image-Loader

2.3.2 *Picasso*

Picasso adalah sebuah *library* gambar untuk *Android*. Itu dibuat dan dikelola oleh *Square*, dan melayani pemuatan dan pengolahan gambar. *Picasso* menyederhanakan proses menampilkan gambar dari lokasi eksternal. Dalam banyak kasus hanya beberapa baris kode yang diperlukan untuk mengimplementasikan penggunaan *library* ini.

Picasso sangat baik untuk menampilkan gambar jarak jauh (dari *server*). *Library* ini menangani setiap tahap proses, dari permintaan *HTTP* awal untuk

caching gambar (Tutplus, 2014). Banyak hal yang bisa ditangani dengan baik dalam *imageloading* pada *android* oleh *Picasso*, yaitu:

1. Penanganan *ImageView recycling* dan pembatalan download pada *adapter.*
2. Melalukan transformasi gambar yang komples dengan penggunaan *memory* yang minimal.
3. Otomatisai dalam *memory dan disk caching* (*Picasso Github*, 2013).

Halaman mengenai *library picasso* dapat diakses pada:

https://github.com/square/*picasso*

2.3.3 *Glide*

Glide adalah *library open source* yang bisa memanajemen pemuatan gambar pada *android* secara cepat dan efisien karena sangat mudah diimplementasikan. *Glide* dapat menangani *decoding* media dan *memory* serta *disk caching*. Fokus utama *Glide* adalah untuk membuat banyak gambar dapat bergulir (*scrolling*) semulus dan secepat mungkin, tapi *Glide* juga efektif untuk hampir semua kasus di mana *developer* biasanya perlu untuk mengambil, mengubah ukuran, dan menampilkan gambar dari *server*.

Glide mendukung pengambilan, *decoding*, dan menampilkan video, gambar, dan animasi *GIF*. *Glide* termasuk *API* yang fleksibel yang memungkinkan pengembang untuk memadukannya dengan berbagai *library* koneksi jaringan. Secara default *Glide* menggunakan *HttpURLConnection*, tetapi juga bisa dipadukan dengan *Google's Volley project* atau *Square's OkHttp library (In the Cheese Factory, 2015)*.

Halaman mengenai *library glide* dapat diakses pada:

https://github.com/bumptech/*glide*

2.4 Konsumsi Daya

Kepopuleran ponsel pintar berbanding lurus dengan kinerjanya semakin baik. Sedangkan besar dan volume kapasitas baterai terbatas, oleh karena itu, teknologi yang hemat energi menjadi penting untuk ponsel pintar (Rong Wu,D., Yi Chen,K., 2014). Keterbatasan energi pada baterai ponsel pintar telah menjadi isu kritis di seluruh dunia. Pada ponsel pintar, kemampuan daya baterai tidak bisa mengimbangi kemajuan teknologi lainnya (misalnya, *CPU* dan *memory*) hal itu dengan cepat menjadi perhatian, terutama dalam konsumsi daya ponsel pintar untuk konten multimedia (Trestian, 2012).

2.4.1 Aplikasi *Trepn profiler*

Trepn profiler adalah aplikasi pemantau energi yang dikembangkan oleh Qualcomm, untuk perangkat *Android* yang menggunakan prosesor *Qualcomm Snapdragon*. *Trepn*dapat membantu pengembang mengoptimalkan penggunaan dan frekuensi *CPU*, statistik *memory*, dan penggunaan jaringan. Ini dapat menampilkan data secara *real-time* atau menyimpannya dalam *file log* untuk

analisis *offline (Qualcomm Developer, 2013)*. *Trepn*menawarkan serangkaian pengaturan yang sangat mudah disesuaikan, yang dapat membantu pembuatan profil real-time berdasarkan penggunaan *CPU*, *memory* atau jaringan. Aplikasi ini juga menawarkan grafik rinci yang dapat membantu dalam menganalisis bagaimana aplikasi menggunakan energi. *Trepn* dapat menganalisa satu aplikasi tertentu, atau perangkat secara keseluruhan.

Gambar 2.1 Gambar Tampilan Aplikasi *Trepn profiler*

(sumber: Qualcomm *Developer*, 2013)

Penggunaan daya pada masing-masing aplikasi *android* terkadang sulit untuk teridentifikasi, karena banyak aplikasi yang berjalan dalam satu waktu. Selain itu masih ada juga *background-process* yang berjalan pada sistem operasi *android*. Untuk mengatasi hal ini, *Trepn profiler* bisa mengidentifikasi konsumsi daya dari masing-masing aplikasi.

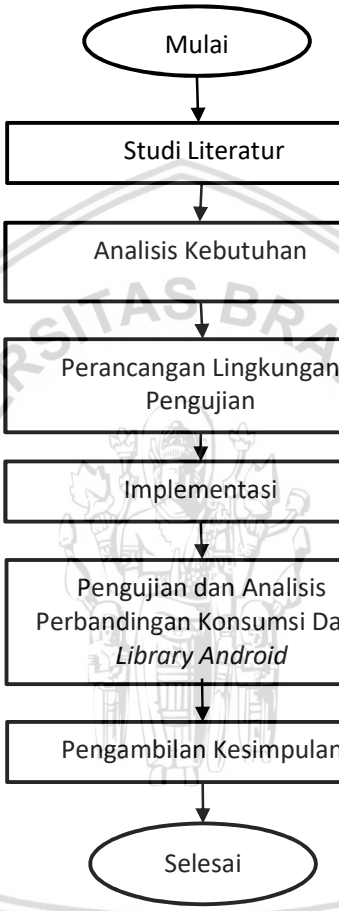
Menu utama dari aplikasi *Trepn profiler* adalah pengukuran konsumsi penggunaan daya secara keseluruhan (sistem), pengukuran konsumsi penggunaan daya tiap aplikasi, dan analisis pengukuran yang telah dilakukan, yaitu melihat kembali statistik dan grafik dari pengukuran yang sudah disimpan sebelumnya, dan terakhir terdapat menu pengaturan yang berisi pengaturan yang bias kita sesuaikan pada aplikasi *Trepn profiler* terkait dengan parameter pengukuran daya yang dibutuhkan dalam penelitian. Langkah-langkah dalam melakukan pengujian konsumsi daya aplikasi menggunakan aplikasi *Trepn profiler* adalah dengan cara mengatur terlebih dahulu pada menu setting untuk mengatur parameter apa yang ingin kita catat dalam mengukur konsumsi daya. Misalkan pada penellitian ini digunakan parameter *memory* dan *CPU* sebagai parameter pendukung dalam pengujian tingkat konsumsi daya sebagai parameter utama.

Pengujian menggunakan aplikasi *Trepn profiler* dapat dipengaruhi oleh berbagai faktor, antara lain: Merk *Smartphone*, *Chipset* yang digunakan, dan *OS*

android yang digunakan. Berbagai faktor itu akan mengakibatkan tingkat manajemen baterai yang berbeda. Namun secara umum pengujian yang disarankan menggunakan Aplikasi *Trepn profiler* adalah *smartphone* dengan *chipset Qualcomm* dan dengan *OS android* tanpa *custom ROM*.

BAB 3 METODOLOGI

Bab ini akan menjelaskan tentang tahapan yang akan dilakukan dalam penelitian analisis perbandingan konsumsi daya *library image loader* pada *android*. Diagram alir pengerjaan penelitian ini ditunjukkan dalam Gambar 3.1.



Gambar 3.1 Gambar Diagram Alir Metodologi Penelitian

3.1 Studi Literatur

Studi literatur merupakan tahapan yang manjadi acuan penulis dalam membuat penelitian ini karena digunakan untuk mendukung dan menunjang pembuatan penelitian ini. Studi literatur berisi tentang penelitian yang sudah ada sebelumnya serta dasar teori yang digunakan dalam penelitian ini. Sumber yang digunakan sebagai literatur berasal dari jurnal, paper serta internet. Teori dan daftar pustaka yang berkaitan dengan penelitian ini meliputi:

1. Kajian Pustaka yang berisi metodologi serta hasil dari penelitian sebelumnya sudah ada.

2. Gambar Digital merupakan objek penelitian yang digunakan dalam pengujian *library,* yang diwujudkan dalam bentuk studi kasus pada aplikasi media sosial.
3. *Library Image loader Android* adalah *library* yang akan diuji pada penelitian ini yaitu *library Universal image loader (UIL), Picasso* dan *Glide.*
4. Konsumsi Daya sebagai fokus utama dalam penelitian ini dan selanjutnya perlu menggunakan Aplikasi *Trepn profiler* sebagai aplikasi tambahan untuk mengukur tingkat konsumsi daya pada sebuah aplikasi *android*.

3.2 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem yang akan digunakan dalam penelitian ini. Analisis kebutuhan terdiri dari tiga hal, yaitu identifikasi aktor, analisis kebutuhan fungsional dan analisis kebutuhan non-fungsional. Analisis kebutuhan fungsional digunakan untuk mengetahui fungsi apa saja yang harus dirancang dalam sistem. Kebutuhan fungsional nantinya akan menjadi sebuah *Use Case* diagram. Sedangkan dalam analisis kebutuhan non-fungsional merupakan proses analisis diluar dari kebutuhan fungsional. Fungsi ini untuk menentukan atribut atau kualitas secara keseluruhan dari sistem.

Pada analisis kebutuhan akan dijelaskan bahwa dalam penelitian ini akan berfokus untuk melakukan pengujian untuk membandingkan tingkat konsumsi daya *library Image loader* pada *android*. Sehingga setiap hal mulai dari hal yang dipersiapkan sebelum pengujian, lingkungan serta langkah pengujian hingga terakhir pada cara membandingkan tingkat konsumsi dayanya dibahas di sini.

Hal yang dibutuhkan pada penelitian ini adalah pembuatan purwarupa aplikasi media sosial yang akan difokuskan sebagai sistem *dummy* untuk mengukur tingkat konsumsi daya *library image loader*. Pembuatan Sistem *dummy* ini adalah sebagai implementasi *library image loader* yang akan diuji tingkat konsumsi dayanya. Purwarupa aplikasi ini akan terhubung dengan *server localhost* yang terhubung melalui jaringan intranet. Basisdata yang terdapat pada *server* untuk menyimpan data media sosial akan diimplementasikan dengan *mysql*. Sementara untuk pengkukuran konsumsi daya digunakan aplikasi pengukuran *Trepn profiler*.

3.3 Perancangan Lingkungan Pengujian

Tahap perancangan lingkungan pengujian dilakukan setelah tahap analisis kebutuhan. Lingkungan pengujian dirancang berdasarkan kebutuhan-kebutuhan yang telah didefinisikan pada tahap analisis kebutuhan. Lingkungan pengujian disini berupa sistem *dummy* purwarupa aplikasi media sosial *instagram*. Terdapat beberapa langkah dalam perancangan lingkungan pengujian yaitu perancangan basis data yang dilakukan pada sisi *server* untuk menyimpan data teks dan gambar, penggunaan *library* yang akan diuji konsumsi dayanya, perancangan *Web API* dan perancangan *screen flow*.

3.3.1 Perancangan Basis Data

Perancangan basis data bertujuan untuk merancang susunan data yang akan digunakan dalam pembuatan aplikasi di sisi *server*. Hasil perancangan tersebut akan diimplementasikan menggunakan *mysql*. *Server* yang digunakan dalam penelitian ini adalah *server* lokal (*localhost*).

3.3.2 Perancangan Web API

Perancangan *Web API* dilakukan setelah perancangangan basis data. Perancangan *Web API* dilakukan agar client dan *server* dapat saling berkomunikasi. Data yang dikirimkan dalam *API* berformat *JSON*. Perancangan dilakukan sehingga sistem *dummy* purwarupa *android* dapat menerima *response* dari *server* sesuai dengan parameter yang diminta. Parameter yang digunakan merupakan Isi data seperti sebuah postingan media sosial pada umumnya yaitu: Nama user, lokasi user, gambar yang diunggah serta *caption* dari sebuah postingan tersebut.

3.3.3 Penggunaan *Library*

Pada tahapan penggunaan *library*, dilakukan pengkondisian pemanggilan dan penggunaan *library image loader* agar ketiga *library* bisa diuji dengan kondisi yang sama. *Library image loader* ini akan menangani gambar yang dikirim oleh *sever* ke aplikasi penelitian ini. *Library image loader* yang digunkan pada penelitian ini adalah *Picasso, glide, dan UIL*. Ketiganya akan diberikan perlakuan dan parameter yang sama pada saat pengujian aplikasi. Perbedaan diusahakan hanya dalam cara pemanggilan dan penggunaan tiap *library* sehingga diharapkan penelitian ini benar-benar bisa membandingkan ketiga *library* dengan kondisi yang sama.

3.3.4 Perancangan *Screen flow*

Perancangan *screen flow* bertujuan untuk memberikan gambaran bagaimana langkah-langkah pada penelitian ini berjalan mulai dari cara bagaimana pengguna mengoperasikan purwarupa aplikasi media sosial tersebut sampai bagaimana cara pengguna bisa melakukan pengukuran konsumsi daya.

3.4 Implementasi

Setelah dilakukan perancangan lingkungan pengujian, maka tahapan selanjutnya adalah implementasi. Implementasi dilakukan dengan membuat sistem *dummy* yang diwujudkan dengan pembuatan purwarupa aplikasi *native android* yang menyerupai aplikasi media sosial. Aplikasi dibuat secara *native* agar aplikasi bekerja secara optimal. Purwarupa aplikasi diimplementasikan dengan bahasa pemrograman *Java* dengan menggunakan *tools* pengembangan *Android Studio*. Purwarupa aplikasi ini dibuat dengan menggunakan *library image loader* yaitu *UIL, Glide dan Picasso*. Sedangkan untuk implementasi *server* menggunakan basisdata *Mysql*. Selanjutnya purwarupa aplikasi dan *server* nantinya saling teringrasi dengan *web service* menggunakan arsitektur *REST* berformat *JSON* dalam proses pengambilan data.

3.5 Pengujian dan Analisis Konsumsi Daya *Library Image loader*

Pengujian dilakukan dengan cara menjalankan serangkaian skenario pengujian terhadap sistem *dummy* aplikasi purwarupa media sosial yang menggunakan *library image loader*. Hal yang dilakukan pertama kali adalah membuka aplikasi pengukuran *Trepn profiler* untuk mulai melakukan proses pengukuran. Setelah itu menjalankan purwarupa aplikasi media sosial *instagram* yang di buat. Setelah itu dipilih *library* yang ingin diuji kemudian memilih ukuran gambar yang diambil dari *server* untuk diuji. Setelah itu *timeline* / linimasa aplikasi akan berjalan secara otomatis *(auto-scrolling)* selama 15 menit, sesuai dengan statistik rata-rata penggunaan aplikasi *instagram* tiap harinya (sosialmediatoday, 2017). Setelah itu hasil pengkuran akan disimpan dalam file berekstensi .csv atau .db. Setiap pengujian dilakukan sebanyak 5 kali untuk masing-masing mendapatkan hasil yang bervariasi dan akurat. Data akan ditampilkan dalam bentuk tabel maupun grafik. Selanjutnya bisa dilakukan analisis terkait perbandingan konsumsi daya dari setiap *library* dan pengaruh ukuran gambar pada pengujian sesuai dengan rumusan masalah pada penelitian ini.

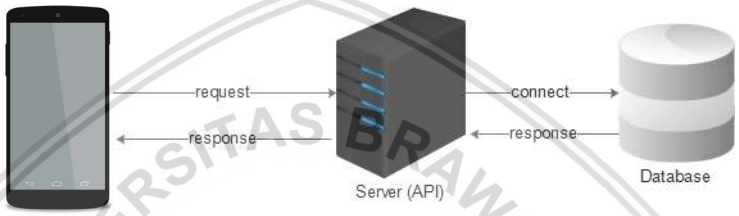
3.6 Pengambilan Kesimpulan

Pengambilan kesimpulan dapat dilakukan setelah melakukan semua tahapan pada penelitian ini. Mulai dari studi literatur, analisis kebutuhan, perancangan lingkungan pengujian, implementasi, pengujian, hingga analisis konsumsi daya *library image loader*. Kesimpulan harus bisa menjawab berbagai rumusan masalah pada penelitian ini yaitu bagaimana cara membandingkan konsumsi daya dari setiap *library picasso, glide dan UIL*, bagaimana hasil perbandingan konsumsi daya pada *library picasso, glide dan UIL* pada studi kasus aplikasi media sosial berbasis *android*, dan bagaimana pengaruh perbedaan ukuran gambar terhadap *library picasso, glide dan UIL* yang diuji. Selain kesimpulan, terdapat juga saran yang bisa digunakan peneliti selanjutnya sebagai acuan dan pertimbangan untuk pengembangan penelitian maupun untuk penelitian yang lain.

BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN

4.1 Deskripsi Umum Sistem

Analisis perbandingan konsumsi daya *library image loader* ini dilakukan dengan cara membuat sistem *dummy* purwarupa aplikasi media sosial yang menyerupai *instagram* dan memanfaatkan *library image loader* dalam menangani gambarnya. Aplikasi tersebut memperoleh data dari basis data yang ada pada *server* lokal secara intranet.



Gambar 4.1 Skenario pertukaran data

1. *Client* melakukan *request data* ke *server*. Isi dari request data berupa pilihan *library* yang ingin diuji.
2. *Server* menerima *request* dari *client* kemudian akan melakukan koneksi ke *database*. *Server* yang digunakan merupakan *server* lokal (*localhost)* agar kondisi pengujian tidak terganggu dengan jaringan luar.
3. Database akan mengirimkan *response* berupa data yang diminta oleh *web service*.
4. Jika database tidak ditemukan maka *web service* akan mengirimkan pesan error ke *client*. Namun jika database ditemukan maka *web service* akan melakukan memberikan data tersebut ke dalam bentuk *JSON* dikirimkan ke *client*.

Perancangan pada bab ini dibagi dalam beberapa bagian yaitu perancangan antarmuka, perancangan basis data, perancangan *web service,* perancangan *library image loader* , dan perancangan skenario pengujian.

4.2 Kebutuhan Sistem

Kebutuhan bertujuan untuk mengetahui kebutuhan yang diperlukan dalam membangun sistem ini. Kebutuhan sistem pada penelitian ini terbagi menjadi dua yaitu kebutuhan fungsional dan non fungsional.

4.2.1 Kebutuhan Fungsional

Kebutuhan fungsional berisi tentang proses-proses apa saja yang harus dapat dilakukan oleh sistem. Kebutuhan fungsional pada sistem ini adalah purwarupa Aplikasi dapat menerima data gambar dari *server* dan dapat berjalan secara otomatis selama 15 menit tanpa mengalami *force close.*

4.2.2 Kebutuhan Non Fungsional

Kebutuhan non fungsional dari sistem yang akan dibangun yaitu:

1. *Usability*

   Tampilan aplikasi dibuat mirip dengan aplikasi *media sosial instagram*
2. *Portability*

   Karena sistem yang akan dibuat bertujuan untuk dilakukan mengujian maka sistem hanya mampu berjalan pada jaringan lokal
3. *Supportability*

   Aplikasi dapat dijalankan pada sistem operasi *Android* minimal versi 4.1 Jelly Bean (API 16).

4.3 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk menentuka semua kebutuhan yang diperlukan untuk membangun sustem perangkat lunak. Analisis juga dibutuhkan untuk menentukan lingkungan pengujian agar pengujian valid dan akurat. Berikut ini dijelaskan tentang Analisi Kebutuhan Sistem yang terdiri dari Identifikasi Aktor, Daftar Kebutuhan Sistem, Diagram *Use Case*, Skenario *Use Case* dan Diagram *Activity*.

Penggunaan *library* bertujuan untuk merancang urutan tahapan yang digunakan pada implementasi sistem. *Library image loader* ini akan menangani gambar yang dikirim oleh sever ke purwaruapa aplikasi media sosial. *Library Image Loader* yang digunkan pada penelitian ini adalah *Picasso, Glide, dan UIL*. Ketiganya akan diberikan perlakuan dan parameter yang sama pada saat pengujian aplikasi. Perbedaan diusahakan hanya dalam cara pemanggilan dan penggunaan tiap *library* sehingga diharapkan penelitian ini benar-benar bisa membandingkan ketiga *library* dengan kondisi yang sama.

Sebelum melakukan pengujian, maka nanti akan disebutkan spesifikasi perangkat yang digunakan pada pengujian kali ini beserta dengan batasan implementasinya. Untuk berfokus pada cara membandingkan tingkat konsumsi daya, maka pada saat pengujian, akan diberikan skenario pengujian yang terdiri dari kondisi lingkungan pengujian serta langkah-langkah pengujian. Setelah didapatkan hasil pengukuran konsumsi daya masing-masing *library* maka akan dicatat serta dianalisis. Pada saat menganalisis hasil pengukuran, hasil tersebut dibandingkan dengan hasil pengukuran kinerja *memory* serta *CPU* sebagai penguat analisis hasil pengujian.

4.3.1 Identifikasi Aktor

Sistem yang akan dibuat hanya memiliki satu aktor yaitu *user*. *User* adalah pengguna sistem *dummy* yang berupa purwarupa aplikasi *instagram*. *User* akan mengirimkan dan memilih menu sesuai kebutuhan pengujian, kemudian aplikasi akan mengirimkan *request* kepada *web API.*

4.3.2 Daftar Kebutuhan Fungsional

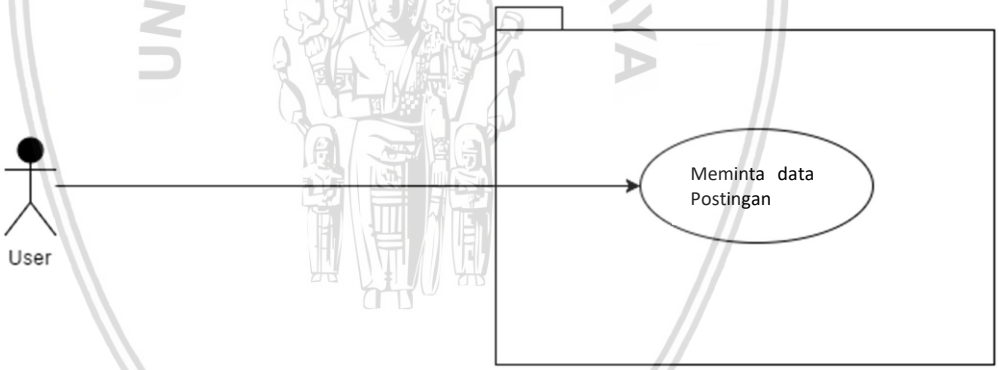
Tabel daftar kebutuhan dapat dilihat pada Tabel 4.1.

Tabel 4.1 Tabel Kebutuhan Fungsional

Identifikasi	Kebutuhan	*Use Case*
FU-01	Pengguna mengirimkan *request* / permintaan data postingan sesuai parameter pengujian yang dipilih.	Meminta data postingan

4.3.3 Diagram *Use Case*

Pemodelan diagram *Use Case* sistem diperoleh dari kebutuhan fungsional



Gambar 4.2 Gambar diagram *Use Case* Meminta data postingan

Gambar 4.2 merupakan *Use Case* sistem yang akan dibuat. Pada Gambar 4.2 tersebut digambarkan *user* dapat mengirimkan *request* data kemudian *Web API* akan memberikan respon data sesuai dengan yang telah di *request*.

4.3.4 Skenario *Use Case*

Bagian skenario *Use Case* berisi nama *Use Case*, aktor yang menjalankan *Use Case*, tujuan dari *Use Case,* deskripsi mengenai *Use Case*, kondisi awal yang harus dipenuhi dan kondisi akhir yang diharapkan setelah berjalannya fungsional *Use Case*. Selain itu terdapat pula alternatif flow sebagai kondisi jika kondisi awal dan normal flow tidak terpenuhi.

Tabel 4.2 Deskripsi *Use Case meminta data* postingan

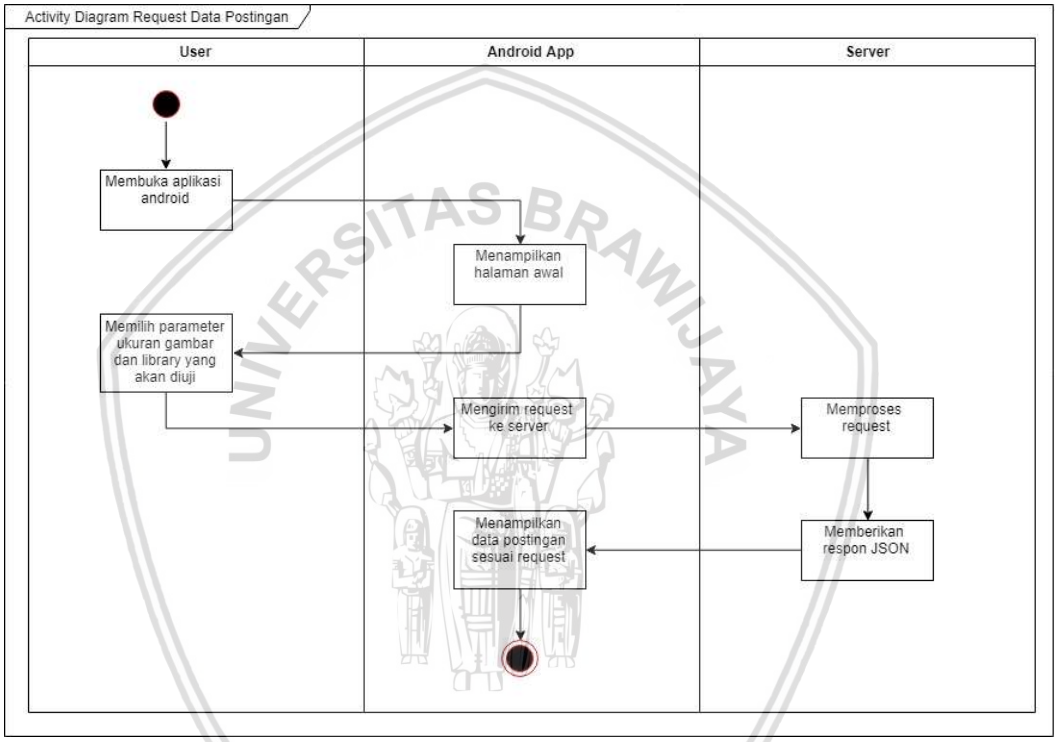
Identifikasi	
Nama	Meminta data postingan
Deskripsi	*Use Case* ini menjelaskan aktor dapat mengirim permintaan data postingan ke *server*. Data postingan merupakan data yang disimpan pada *database server* yang berisi data user, lokasi, gambar serta *caption* dari postingan media sosial.
Aktor	User
Kondisi Awal	Aktor telah membuka sistem *dummy* aplikasi purwarupa media sosial dan telah terhubung ke *server* melalui jaringan lokal. Aplikasi akan langsung menampilkan halaman yang menampilkan parameter ukuran gambar serta *library* yang bisa dipilih oleh user dalam melakukan pengujian.
Kondisi Akhir	Sistem mengirimkan data postingan media sosial kepada *user* sesuai dengan parameter yang dipilih.
Normal Flow	Permintaan dari *user* akan dikirim ke *server* lalu *server* memberikan respon sesuai parameter dari permintaan *user*. Saat melakukan permintaan, user dapat memilih 2 parameter yang disediakan sistem, yaitu parameter ukuran gambar dan *library* yang digunakan. Parameter ukuran gambar digunakan untuk menguji pengaruh ketiga parameter *library image loader* yang diuji, terhadap berbagai ukuran gambar yang akan diminta dari *server*.
Alternatif Flow	Jika user sedang tidak terhubung dengan *server* maka permintaan tidak akan dikirimkan, menunggu sampai terhubung dengan *server*

4.3.5 Diagram *Activity*

Perancangan *Activity* diagram dilakukan untuk menggambarkan detail alur aktivitas-aktivitas yang terjadi pada aplikasi dan sistem yang sedang dibangun. Alur aktivitas yang dijabarkan dalam bentuk *Activity diagram* adalah aktivitas yang telah didefinisikan pada skenario *Use Case.*

Pada gambar 4.3 yang berisi *diagram Activity* sistem dapat dilihat bahwa ada tiga ruang berbeda yang ada pada sistem, yaitu: user, *android app* (purwarupa aplikasi) serta *server*. Pada awalnya user akan membuka aplikasi, dengan syarat sudah terhubung dengan *server* melalui jaringan lokal. Selanjutnya, aplikasi akan menampilkan halaman awal berupa parameter yang bisa dipilih oleh *user* sebelum melakukan pengujian. Selanjutnya user dapat memilih parameter ukuran gambar serta *library* yang akan diminta untuk diuji. Pada parameter ukuran gambar, terdapat tiga pilihan yang dapat dipilih, yaitu gambar dengan ukuran 100kB, 1MB

dan 3MB, sementara untuk parameter *library* ada tiga pilihan yaitu *library UIL, glide dan Picasso*. Setelah itu aplikasi akan mengirim *request* ke *server* sesuai parameter yang dipilih user. Lalu request tersebut akan diproses oleh *server*. Selanjutnya akan mengirimkan *response* data berupa *JSON* yang nantinya akan ditampilkan pada aplikasi *android*. Aktivitas mulai dari user memilih parameter sampai *android* app menampilkan data postingan dilakukan berulang kali sesuai dengan pengujian yang diperlukan pada penelitian. Pengujian pada penelitian kali ini dilakukan sebanyak 5 kali pengujian untuk tiap ukuran gambar dan tiap *library*. Sementara untuk pengujiannya dilakukan secara bergantian satu-persatu menggunakan satu *smartphone.*



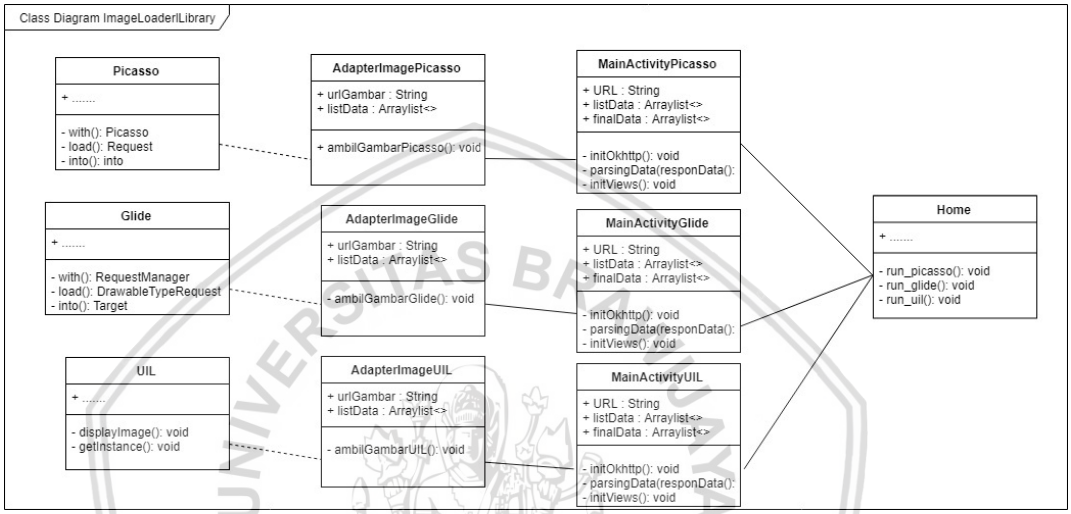
Gambar 4.3 Diagram Aktivitas Sistem

Gambar 4.3 adalah diagram aktivitas sistem yang dibuat untuk menjelaskan interaksi pengguna dengan sistem.

4.3.6 Diagram Class

Perancangan *class* diagram dilakukan untuk menggambarkan kelas beserta relasinya yang terdapat pada purwarupa aplikasi *instagram*. *class diagram* cukup untuk menjelaskan *class* atau *method* apa yang perlu dipanggil dari 3 *library* tersebut. Secara umum method yang diambil dari masing-masing *library* adalah method *load()* pada *picasso dan glide*, sementara pada *UIL* menggunakan *method displayImage().* Sementara Untuk Implementasi *Class* pada aplikasi *android* masing-masing *library* dibuatkan *class adapter* dan *main* masing-masing untuk mempermudah pemanggilan.

Pada *Class main* terdapat *variable URL* yang berisi alamat *IP server* lokal yang dipakai serta dua variable *arraylist<>* yaitu listdata dan finaldata untuk menyimpan sementara dan menampilkan data dari *JSON* yang dikirm oleh *server*. Pada *class adapter* terdapat penggunaan *library image loader* yang diimplementasikan pada method *ambilgambar()* dan *class* ini memilki variable URL yang berisi alamat IP *server* lokal yang dipakai serta variable arraylist<> yaitu untuk menyimpan data dari *JSON* yang dikirm oleh *server*.



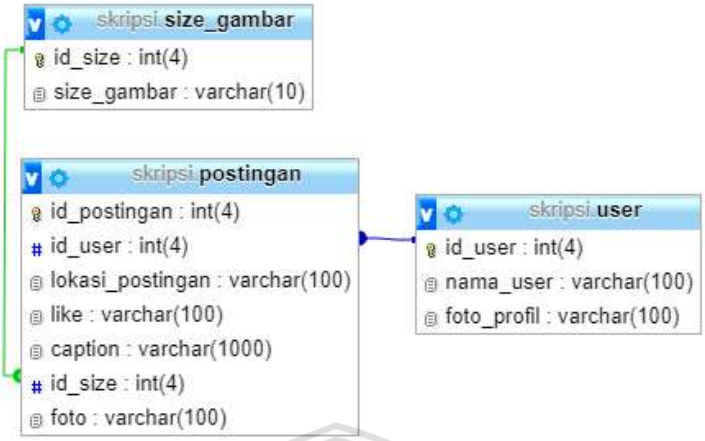
Gambar 4.4 Gambar Diagram *Class* request data postingan

Gambar 4.4 adalah diagram *class* yang berisi variable, *method* dan *class* yang digunakan pada sistem.

4.3.7 Perancangan Basis Data

Setelah melakukan perancangan *class* diagram, dilakukan perancangan basis data. Perancangan basis data bertujuan untuk merancang susunan data yang akan digunakan dalam pembuatan aplikasi di sisi *server*. Perancangan akan dibuat dalam bentuk tabel dan *Entity Relational Diagram (ERD).* Hasil perancangan tersebut akan diimplementasikan menggunakan *mysql*.

Basisdata yang dipakai bernama db_skripsi dan memiliki tiga tabel, yaitu tabel user, tabel postingan dan tabel size_gambar dengan bentuk ERD seperti berikut ini:



Gambar 4.5 ERD Database db_skripsi

Pada tabel user berisi data user yang meliputi id_user, nama_user serta foto_profil user. Sementara pada tabel postingan, berisi data postingan media sosial yang meliputi id_postingan, id_user, lokasi, like, foto dan caption sebuah postingan. Pada tabel postingan juga terdapat kolom id_size yang nantinya akan digunakan untuk membedakan ukuran gambar sesuai dengan ukuran yang dibutuhkan dalam pengujian. Selanjutnya pada tabel size_gambar terdapat terdapat kolom id_size, dan size_gambar yang berisi klasifikasi ukuran gambar sesuai yang dibutuhkan pada pengujian yaitu sebesar 100kB, 1MB dan 3MB.

4.3.8 Perancangan *Web API*

Server yang digunakan dalam penelitian ini adalah *server* local (*localhost*). Sementara perancangan *Web API* dilakukan agar client dan *server* dapat saling berkomunikasi. *Framework* yang digunakan pada implementasi *Web API* adalah *CodeIgniter(CI)*. Data yang dikirimkan dalam *API* berformat *JSON*. Isi data yang dikirim berupa isi dari sebuah postingan pada umumnya yaitu data user, lokasi, gambar serta *caption* dari postingan media sosial yang nantinya akan diimplementasikan dalam sebuah fungsi *get* yang mengambil datanya pada *database server.*

4.3.9 Penggunaan *Library*

Penggunaan *library image loader* akan diimplementasikan pada sebuah fungsi yang akan digunakan untuk memanggil dan menampilkan gambar secara *default*. Sehingga ketiga *library* mendapatkan kondisi yang sama ketika diuji. Tujuan pengkondisian *library Image loader* secara default agar mengetahui bagaimana sebuah *library* menangani pemanggilan gambar secara default dan membandingkannya dengan *library* lainnya.

4.3.10 Perancangan Screen Flow

Perancangan *screen flow* merupakan tahap perancangan alur-alur perpindahan halaman pada penelitian ini. Perancangan *screen flow* bertujuan

untuk memberikan gambaran bagaimana langkah-langkah pada penelitian ini berjalan mulai dari cara pengukuran konsumsi daya sampai bagaimana pengguna mengoperasikan purwarupa aplikasi media sosial tersebut. Sementara perancangan desain antar muka purwarupa aplikasi dibuat mirip aplikasi media sosial *instagram*, karena *Instagram* adalah aplikasi untuk berbagi foto/gambar paling popular di dunia saat ini (Sarungpreneur, 2015). Pada *screen flow* nantinya akan diketahui bagaimana alur pengujian mulai dari cara pengukuran konsumsi daya dan cara mencatatnya serta ada tahapan pemilihan parameter dalam sebuah pengujian yang dapat dipilih oleh user.

BAB 5 IMPLEMENTASI

Bab ini akan berisi implementasi dari perancangan yang telah dibahas sebelumnya untuk mewujudkan sistem yang dapat melakukan pengujian sesuai analisa kebutuhan

5.1 Spesifikasi Perangkat Keras

Dalam penelitian ini, peneliti menggunakan PC berjenis notebook dengan spesifikasi yang ditunjukkan pada Tabel 5.1. Sedangkan spesifikasi perangkat keras pada *smartphone* ditunjukkan pada Tabel 5.2.

Tabel 5.1 Spesifikasi Perangkat Keras Komputer

Nama Komponen	Spesifikasi
Prosesor PC	Intel Core i3 3217U , 1,80GHz
Memory (RAM) PC	8.00 GB
Harddisk PC	500 GB

Tabel 5.2 Spesifikasi Perangkat Keras *Smartphone*

Nama Komponen	Spesifikasi Perangkat Keras
Prosesor *Smartphone*	Octa-core
Chipset *Smartphone*	Qualcomm MSM8994 Snapdragon 810
Memory (RAM) *Smartphone*	3 GB
Kapasitas Baterai	3000 mA

5.2 Spesifikasi Perangkat Lunak

Berikut adalah spesifikasi Perangkat lunak yang digunakan peneliti untuk membantu penelitian:

Tabel 5.3 Spesifikasi Perangkat Lunak Komputer

Nama	Spesifikasi
Sistem Operasi	Microsoft Windows 10
Tools Pemrograman	*Android* Studio 2.2.2, Sublime 2.0.2
Bahasa Pemrograman	PHP 5, Java
DBMS	*Mysql* 5.6
Web *Server*	Apache 2.4.4

Tabel 5.4 Spesifikasi Perangkat Lunak *Smartphone*

Nama	Spesifikasi
Sistem Operasi	*Android* Marsmellow 6.1
Versi Kernel	3.10.84-perf-gf1cfa5f
Profiler App	*Trepn profiler* 6.2s

5.3 Batasan Implementasi

Batasan-batasan implementasi sistem ini sebagai berikut:

1. Fokus penelitian terdapat pada pengujian dan perbandingan *library*. Sehingga purwarupa aplikasi media sosial dan aplikasi pengukur daya hanya sebagai lingkungan pengujian.
2. Pengujian dilakukan dengan menggunakan jaringan lokal guna mengurangi gangguan yang disebabkan oleh koneksi jaringan.
3. Ketiga *library* akan diberi perlakuaan yang sama, yakni dengan menggunakan pengaturan *default* dalam memuat gambar.

5.4 Implementasi Basis Data

Implementasi Basis data yang dipakai dalam penelitian ini dibuat menggunakan Query *Mysql*. Basisdata yang digunakan bernama db_skripsi dan mempunyai 2 tabel, yaitu tabel user dan tabel postingan.

5.4.1 Implementasi tabel User

Tabel 5.5 Implementasi Kode *Query SQL tabel user*

```
1. CREATE TABLE IF NOT EXISTS `user` (
2.   `id_user` int(2) NOT NULL AUTO_INCREMENT,
3.   `nama_user` text NOT NULL,
4.   `foto_profil` text NOT NULL,
5.   PRIMARY KEY (`id_user`)
6. )  PRIMARY KEY (`id`)
7. ) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=15 ;
```

Tabel pertama adalah tabel user. Tabel ini memiliki tiga kolom yaitu kolom id_user, nama_user dan foto_profil. Id_user bertipe *integer* dan bersifat *auto increment*. Sementara nama_user dan foto_profil bersifat *text*. Pada foto_profil nantinya akan berisi dimana direktori dan nama foto profil itu disimpan.

5.4.2 Implementasi tabel Postingan

Tabel 5.6 Implementasi Kode *Query SQL tabel postingan*

```
8. CREATE TABLE IF NOT EXISTS `postingan` (
9.   `id_postingan` int(4) NOT NULL AUTO_INCREMENT,
10.  `id_user` int(2) NOT NULL,
11.  `lokasi_user` text NOT NULL,
12.  `like` text NOT NULL,
13.  `status_user` text NOT NULL,
14.  `foto` text NOT NULL,
15.  PRIMARY KEY (`id_postingan`)
16.)  PRIMARY KEY (`id`)
17.      ) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=15
   ;
```

Tabel kedua adalah tabel postingan. Pada tabel ini terdapat lima kolom, yaitu id_postingan, id_user, lokasi_user, like, status_user dan foto. Id_postingan

dan Id_user bertipe integer, sementara lokasi_user, like, status_user dan foto bertipe text. Id_postingan bersifat *auto increment*, sementara id_user mengambil dari tabel user dengan menggunakan *foreign key*. Pada foto nantinya akan berisi dimana direktori dan nama foto profil itu disimpan.

5.4.3 Implementasi tabel Size Gambar

Tabel 5.7 Implementasi Kode *Query SQL* Size Gambar

```
1. CREATE TABLE IF NOT EXISTS `size_gambar` (
2.   `id_size` int(4) NOT NULL,
3.   `size_gambar` varchar(10) NOT NULL,
4.   PRIMARY KEY (`id_size`)
5. );
```

Selanjutnya adalah implementasi tabel size gambar. Tabel ini memiliki dua kolom yaitu kolom id_size dan size gambar. Kolom Id size nantinya akan dijadikan *foreign key* dan dipanggil pada tabel postingan untuk mengelompokkan gambar yang disimpan pada tabel postingan. Pengelompokkan dimaksudkan untuk mempermudah pemanggilan saat pengujian.

5.5 Implementasi WEB API

Tabel 5.8 Implementasi WEB API

```
1. function request_data_get($size = 2){
2.         $hasil  = $this->db->query("select * from postingan,
   user where user.id_user = postingan.id_user and
   postingan.id_size = ".$size)->result();
3.             if($hasil){
4.                 $this->response($hasil, 200);
5.             }else{
6.                 $this->response(NULL, 404);
7.             }
8.     }
```

Implementasi *WEB API* diwujudkan dalam sebuah fungsi *get* yang akan memanggil data yang dibutuhkan pada *database server* untuk pengujian. Data tersebut adalah data postingan media sosial sesuai parameter size/ukuran gambar yang dipilih. Parameter ukuran gambar yang dapat dipilih adalah gambar dengan ukuran 100KB, 1MB dan 3MB. Data akan dimasukkan pada variable $hasil. Jika data ditemukan maka akan ditampikan, sementara jika tidak ditemukan makan akan menampilkan pesan data error tidak ditemukan / *not found.*

5.6 Penggunaan *Library*

Implementasi pada penggunaan *library image loader* akan dimasukkan pada sebuah fungsi yang digunakan untuk memanggil dan mengkonfigurasi gambar secara *default*. Sehingga ketiga *library* mendapatkan kondisi yang sama ketika diuji.

5.6.1 Penggunaan *Library Glide*

Penggunaan kode pada pemanggilan *library Glide* ditunjukkan dalam Kode 5.9

Tabel 5.9 Penggunaan Kode *Library Glide*

```
1.      public void onBindViewHolder(ViewHolder viewHolder, int i)
    {
2.
    Glide.with(context).load(urlGambar+listData.get(i).getFoto_prof
    il()).into(viewHolder.foto_profil);
3.
    Glide.with(context).load(urlGambar+listData.get(i).getFoto()).i
    nto(viewHolder.foto);
4.      }
```

Pada tabel 5.9 terdapat pemanggilan *class glide* dengan memanggil *method load()* yang memiliki parameter berupa *url link* gambar yang akan diakses dan dimasukkan pada *imageview* dengan nama foto dan foto profil.

5.6.2 Penggunaan *Library Picasso*

Penggunaan kode *request* pada *Library Picasso* ditunjukkan pada Kode 5.10.

Tabel 5.10 Penggunaan Kode *Library Picasso*

```
1.      public void onBindViewHolder(ViewHolder viewHolder, int i) {

2.  Picasso.with(context).load(urlGambar+listData.get(i).getFoto_pro
    fil()).into(viewHolder.foto_profil);
3.
    Picasso.with(context).load(urlGambar+listData.get(i).getFoto()).
    fit().into(viewHolder.foto);


4.      }
```

Pada tabel 5.10 pemanggilan *class picasso* sama dengan pemanggilan *class glide* pada tabel 5.9, yaitu dengan memanggil *method load()* yang memiliki parameter berupa *url link* gambar yang akan diakses dan dimasukkan pada *imageview* dengan nama foto dan foto profil.

5.6.3 Penggunaan *Library UIL*

Penggunaan kode *request pada Library UIL* ditunjukkan pada Kode 5.11.

Tabel 5.11 Penggunaan *Library UIL*

```
1.  Image loader Configuration config = new Image loader
    Configuration.Builder(context)
2.                  .build();
3.          Image loader .getInstance().init(config);
4.          // END - UNIVERSAL IMAGE LOADER SETUP
5.
6.          Image loader image loader = Image loader .getInstance();
7.          image loader
    .displayImage(urlGambar+listData.get(i).getFoto_profil(),
    viewHolder.foto_profil);
```

```
8. image loader .displayImage(urlGambar+listData.get(i).getFoto(),
   viewHolder.foto);
```

Pada tabel 5.11 dilakukan konfigurasi terlebih dahulu dengan menginstansiasi *class image loader* dengan memanggil method *getinstance().* Lalu pemanggilan method *displayimage()* dengan parameter berupa *url link* gambar yang akan diakses dan dimasukkan pada *imageview* dengan nama foto dan foto profil.

5.6.4 Implementasi *parsing JSON object*

Tabel 5.12 Implementasi *JSON Parsing* ke model data

```
1.   OkHttpClient client = new OkHttpClient();
2.         Request request = new Request.Builder()
3.                 .url(URL)
4.                 .build();
5.
6.         client.newCall(request).enqueue(new Callback() {
7.             @Override
8.             public void onFailure(Call call, IOException e) {
9.                 Log.d("Can't connect to server","");
10.            }
11.
12.            @Override
13.            public void onResponse(Call call, final Response response)
   throws IOException {
14.                final String responseData = response.body().string();
15.                MainActivityGlide.this.runOnUiThread(new Runnable() {
16.                    @Override
17.                    public void run() {
18.                        try{
19.                            result = new JSONArray(responseData);
20.                            Log.d("result", result+"");
21.                            ArrayList listData = new ArrayList<>();
22.
23.                            for(int i=0;i<result.length();i++) {
24.                                JSONObject data =
   result.getJSONObject(i);
25.
26.                                ModelFoto modelData = new ModelFoto();
27.
28.
   modelData.setUser(data.getString("nama_user"));
29.
   modelData.setFoto_profil(data.getString("foto_profil"));
30.
   modelData.setFoto(data.getString("foto"));
31.
32.
   modelData.setLokasi(data.getString("lokasi_postingan"));
33.
   modelData.setLike(data.getString("like"));
34.
   modelData.setStatus(data.getString("caption"));
35.
36.
37.                                listData.add(modelData);
38.                            }
39.                            finalData = listData;
40.                            listData2 = listData;
41.
```

```
42.                         }catch (JSONException e){
43.
44.                         }
45.                     }
46.                 });
47.
48.             }
49.                 });
```

Pada tabel 5.12 adalah mekanisme *parsing JSON* untuk data yang dikirim, yaitu data postingan. Di sini diimplementasikan menggunakan *library okhttpclient* untuk *request http*nya. Pertama dikonfigurasi dulu *request* dan *builder*nya. *Okhttp* berjalan secara *Asynchronus* sehingga memerlukan antrian (*enqueue*), sehingga *method default newcall()* dipanggil dengan parameter *request* yang sudah dikonfigurasi diawal tadi, dan diantrikan dengan *method enqueue*. Hasil pada *request* tadi ditangani 2 *method*, jika berhasil ditangani *onresponse()* , jika gagal ditangani *onfailure().*

Pada *method onresponse()* respon berupa *String* masuk ke kelas *respon.body()*, setelah itu dimasukkan pada *variable respondata*. Lalu dibuat variable *JSONobject* bernama data. Variable data ini tadi diisi melalui mekanisme perulangan yang isinya didapatkan dari *array result* . *Array result* dilooping dan dimasukan pada data dan dimasukkan pada kelas modelfoto yang diinstansiasi melaui modelData dan disesuaikan dengan nama variabel dan *method set* dan *get* nya, yaitu variabel nama_user, foto, foto_profil, lokasi_postingan, like dan caption. Modeldata tadi dimasukkan pada *arraylist* finaldata dan listdata2. *Arraylist* yang dipersiapkan ada 2 data bisa ditambahkan dalam mekanisme perulangan.

5.6.5 Implementasi kode *auto-scrolling* pada *timeline*

Tabel 5.13 Implementasi Kode Auto Scrolling

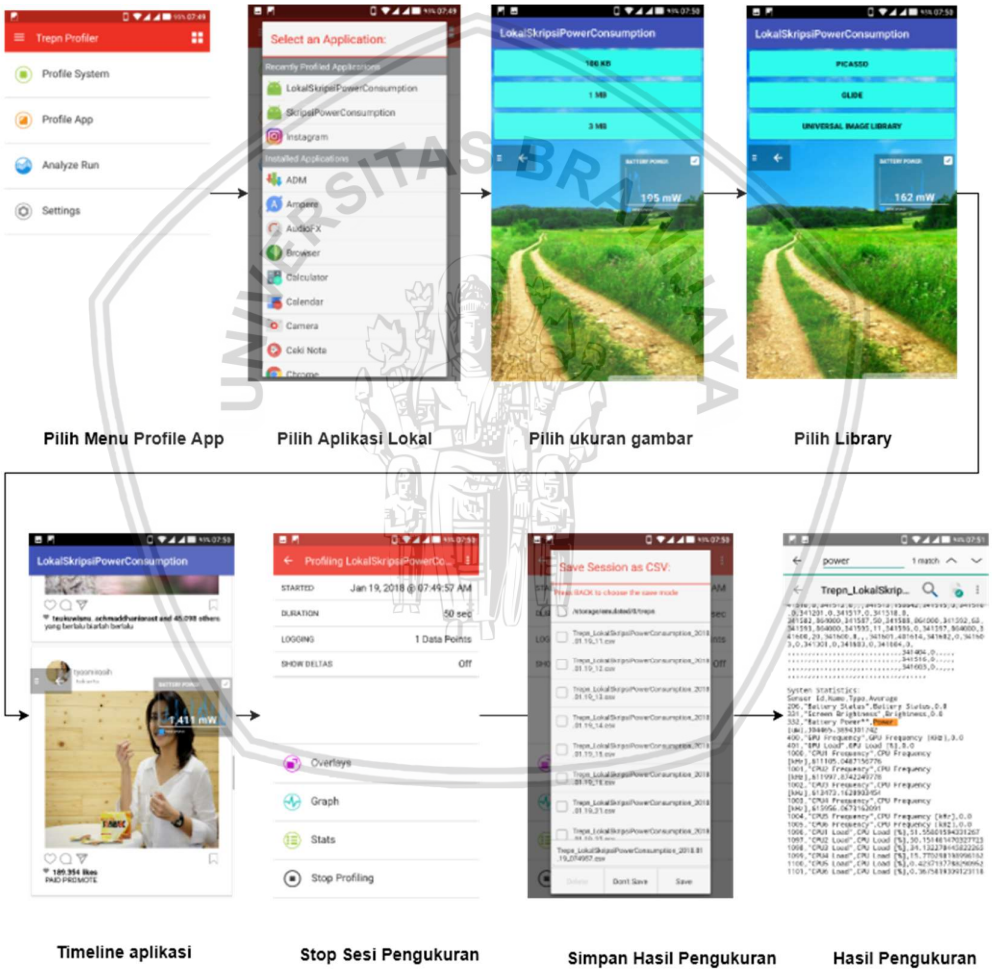
```
1.  handler.postDelayed(new Runnable() {
2.              public void run() {
3.                   initViews();
4.      new CountDownTimer(900000, 2000) {
5.                       int count = 0;
6.           public void onTick(long millisUntilFinished) {
7.
    recyclerView.smoothScrollToPosition(count++);
8.                                              }
9.            public void onFinish() {
10.
    Toast.makeText(MainActivityPicasso.this, "Pengujian Selesai..",
    Toast.LENGTH_LONG).show();
11.                                             }
12.                       }.start();
13.                                   pd.dismiss();
14.                                 }
15.                             },2000
16.              );
```

Pada tabel 5.13 dilakukan pemanggilan method *postdelayed()* pada *handler* yang berfungsi untuk melakukan *scrolling* selama 900.000 *ms* atau 15 menit tiap

2000 *ms* atau 2 detik. Pengkondisian a*uto scrolling* dilakukan oleh *method smoothScrollToPosition()* yang dipanggil di dalam method *onTick().* Setelah 15 menit maka method *onFinish()* akan dipanggil sehingga *scrolling* akan berhenti dan menampilkan tulisan "pengujian selesai" pada layar.

5.7 Implementasi *Screen flow*

Sesuai dengan kebutuhan sistem serta analisis kebutuhan yang telah dibuat, maka diimplementasikan sebuah *screen flow* yang akan menjelaskan alur pengujian mulai dari langkah awal pengukuran konsumsi daya sampai pencatatannya seperti pada gambar 5.1. Sementara untuk detail pengujian akan dijelaskan pada bab selanjutnya pada bagian skenario pengujian.



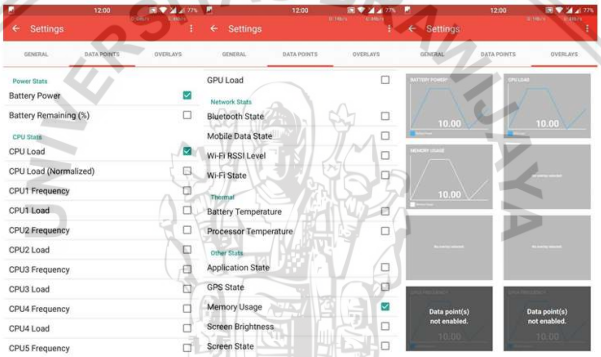
Gambar 5.1 Implementasi Screenflow

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini akan menjelaskan mengenai hal-hal terkait langkah-langkah pengujian dan analisis berdasarkan hasil pengujian.

6.1 Aplikasi Pengukur Konsumsi Daya *Trepn profiler*

Pada penelitian ini, pengukuran konsumsi daya sistem *dummy* purwarupa aplikasi media sosial dilakukan oleh aplikasi *Trepn profiler*. Pada penelitian ini, parameter pendukung yang dipakai dalam mengukur konsumsi daya *library image loader android* ini adalah kinerja *memory* dan *CPU*, sehingga pada *Trepn profiler* perlu di atur pada menu *setting.* Parameter yang diukur yaitu: tingkat konsumsi daya, *memory* serta *CPU*. Setelah parameter diatur, baru memilih menu *Profiler App* dan memilih aplikasi yang akan diukur tingkat konsumsi dayanya, dalam penelitian ini aplikasi yang diukur konsumsi dayanya adalah aplikasi *Lokalpowerconsumption*.



Gambar 6.1 Pengaturan Aplikasi *Trepn profiler*

6.2 Skenario Pengujian

Pengujian pada penelitian ini dilakukan untuk mengukur konsumsi daya dari setiap *library* saat menjalankan purwarupa aplikasi media sosial. *Server* yang digunakan dalam pengujian ini yaitu *server localhost* yang terhubung dengan jaringan lokal intranet dengan purwarupa aplikasi media sosial agar pengujian tidak terpengaruh oleh jaringan luar.

Beberapa hal terkait pengkondisian *smartphone* yang harus disiapkan sebelum pengujian agar pengujian menjadi maksimal antara lain:

1. *Smartphone* dikondisikan tidak dalam keadaan *charging* dan kondisi baterai tidak dalam keadaan kritis (di atas 20%).
2. *Smartphone* dikondisikan terkoneksi dengan *server* pengujian secara intranet.
3. Tingkat kecerahan pada *smartphone* dikondisikan tidak adaptif dan di set pada tingkat kecerahan maksimum 100%.

4. Memastikan tidak ada aplikasi yang berjalan pada *recent apps* sebelum melakukan pengujian dan menghapus *recent apps* sebelum melakukan pengujian selanjutnya.

Pengujian ini dilakukan dengan 3 pilihan ukuran *size* gambar agar mengetahui pengaruh perbedaan ukuran gambar terhadap tingkat konsumsi daya. Ketiga ukuran *size* gambar yang digunakan dalam penelitian ini adalah 100kB, 1MB dan 3MB. Besaran ketiga ukuran gambar tersebut diperoleh peneliti dari beberapa pengujian kecil terkait rentang *size* gambar yang terdapat pada beberapa aplikasi media sosial yaitu dengan cara mengunduh gambar dan melihat ukuran *size*nya. Selain itu besaran *size* gambar yang digunakan dipilih agar hasil pengujian menunjukkan hasil yang cukup signifikan untuk tiap *library* yang diuji.

Tiap pengujian dilakukan selama 15 menit, karena sesuai dengan rata-rata penggunaan aplikasi media sosial *instagram* dalam satu hari (Sosialmediatoday, 2017). Pengujian dilakukan sebanyak 5 kali tiap ukuran gambar dan tiap *library*, dan pengukuran konsumsi daya dilakukan dengan aplikasi *Trepn profiler*. Satuan yang digunakan dalam pengujian ini adalah mW(miliWatt).

Untuk langkah pengujian dapat dilihat pada tabel 4.1

Tabel 6.1 Tabel Langkah pengujian konsumsi daya aplikasi media sosial

Langkah ke:	Langkah Pengujian
1	Membuka Aplikasi pengukuran *Trepn profiler* lalu memilih menu Profile App
2	Memilih untuk mengukur konsumsi daya purwarupa aplikasi *instagram* (lokalskripsipowerconsumption)
3	Masuk ke aplikasi purwarupa dan dilakukan pemilihan ukuran gambar yang akan diuji. Ada 3 pilihan ukuran gambar yaitu 100kB, 1MB dan 3MB.
4	Memilih *library image loader* yang akan diuji, yaitu *Picasso*, *Glide* dan *UIL*.
5	Setelah itu *timeline*/linimasa aplikasi akan berjalan secara otomatis *(auto-scrolling)*. Tiap pengujian dilakukan selama 15 menit.
6	Setelah 15 menit, aplikasi akan berhenti berjalan otomatis, lalu user kembali aplikasi *Trepn profiler* untuk mengentikan sesi pengukuran konsumsi daya dengan menekan tombol *Stop Profiling*.
7	Hasil pengkuran akan disimpan dalam file berekstensi .csv atau .db.
8	Hasil pengukuran dapat di analisis, agar lebih mudah dapat langsung mencari dengan *keyword "power"* agar langsung menemukan hasil pengukuran konsumsi daya.

Pengujian *library* dilakukan menggunakan 1 *smartphone* dan langkah pengujian dilakukan satu-persatu secara bergantian tiap ukuran gambar dan tiap *library*. Sehingga rangkaian langkah pengujian pada tabel 4.1 akan dilakukan sebanyak 45 kali (3 ukuran gambar x 3 *library* x 5 pengujian). Langkah pengujian pada tabel 4.1 dapat dijabarkan melalui screenflow skenario pengujian pada gambar 6.2.

Gambar 6.2 Skenario Pengujian

6.3 Komparasi Hasil Pengujian

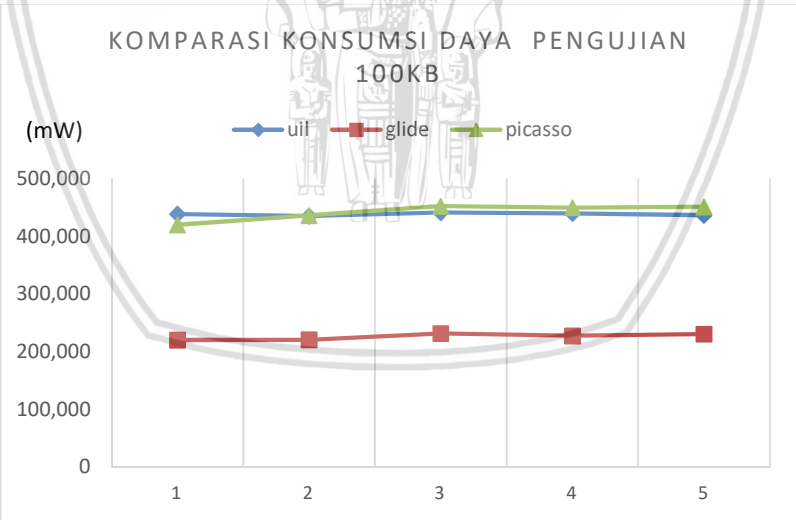
Berikut adalah komparasi rata-rata konsumsi daya tiap *library* (dalam satuan mW/miliwatt) hasil dari pengujian yang telah dilakukan sesuai skenario pengujian.

6.3.1 Komparasi hasil pengujian dengan ukuran gambar 100KB

Tabel 6.2 Nilai hasil pengujian dengan ukuran gambar 100KB (mW)

no	*uil*	*glide*	*picasso*
1	270,505	221,174	312,914
2	266,292	243,718	277,050
3	269,970	227,691	269,182
4	286,319	216,829	275,888
5	272,169	221,935	306,446
Rata2	273,271	227,353	283,758

Pada tabel 6.2 dapat dilihat bahwa hasil pengujian rata-rata konsumsi daya pada masing-masing *library* terhadap gambar berukuran 100kb menunjukkan bahwa *library glide* memiliki konsumsi daya paling rendah yaitu sebesar 227,353mW, sementara *library UIL* dan *Picasso* mengalami hasil yang fluktuatif dan kadang salah satu memiliki tingkat konsumsi daya yang lebih tinggi, dan kadang lebih rendah. Sementara jika dibandingkan dengan *library Glide*, maka kedua *library* tersebut memiliki konsumsi daya dengan nilai keseluruhan diatas konsumsi daya *library Glide*. Secara keseluruhan rata-rata tingkat konsumsi daya paling boros adalah *library Picasso* yaitu sebesar 283,758mW, sementara *library UIL* dengan selisih yang tak begitu jauh berada ditempat kedua dengan rata-rata konsumsi daya sebesar 273,271mW.



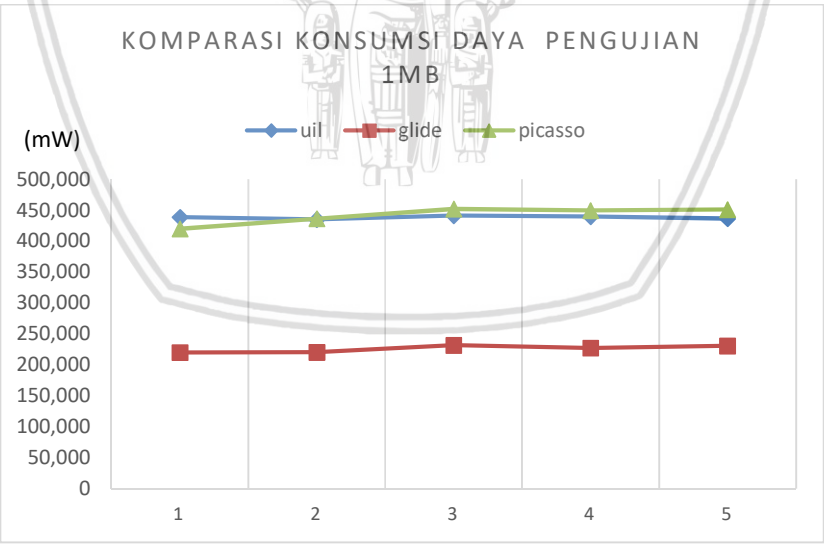
Gambar 6.3 Komparasi Hasil Pengujian Konsumsi Daya 100KB

6.3.2 Komparasi hasil pengujian dengan ukuran gambar 1MB

Tabel 6.3 Nilai hasil pengujian dengan ukuran gambar 1MB (mW)

no	*uil*	*glide*	*picasso*
1	438,889	220,269	420,062
2	435,102	220,889	436,329
3	441,282	231,719	452,184
4	439,945	227,499	449,538
5	436,471	230,985	451,002
Rata2	438,338	226,272	441,823

Pada tabel 6.3 dapat dilihat bahwa hasil pengujian rata-rata konsumsi daya pada masing-masing *library* terhadap gambar berukuran 1MB menunjukkan bahwa *library glide* tetap memiliki konsumsi daya paling rendah yaitu sebesar 226,272mW, sementara *library UIL* dan *Picasso* juga masih mengalami hasil yang fluktuatif dan kadang salah satu memiliki tingkat konsumsi daya yang lebih tinggi, dan kadang lebih rendah. Sementara jika dibandingkan dengan *library Glide*, maka kedua *library* tersebut memiliki konsumsi daya dengan nilai keseluruhan jauh diatas konsumsi daya *library Glide*, bahkan pada rata-rata terlihat kalau konsumsi daya *library UIL* dan *Picasso* hampir dua kali lipat daripada konsumsi daya *library Glide*. Secara keseluruhan rata-rata tingkat konsumsi daya paling boros tetap *library Picasso* yaitu sebesar 441,822mW, sementara *library UIL* ditempat kedua dengan selisih yang tak begitu jauh dengan rata-rata sebesar 438,338mW.



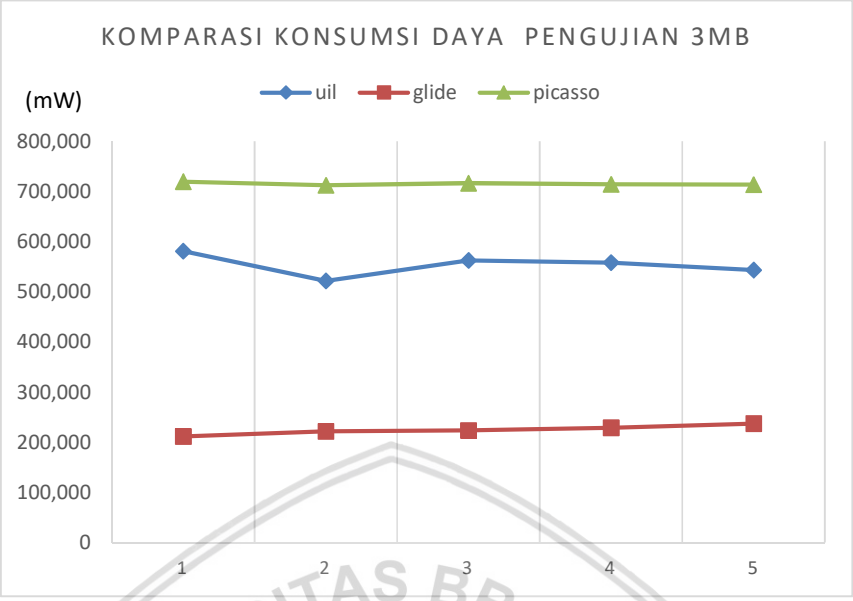
Gambar 6.4 Komparasi Hasil Pengujian Konsumsi Daya 1MB

6.3.3 Komparasi hasil pengujian dengan ukuran gambar 3MB

Tabel 6.4 Nilai hasil pengujian dengan ukuran gambar 3MB (mW)

no	*uil*	*glide*	*picasso*
1	581,030	211,688	719,827
2	521,779	221,707	712,645
3	562,756	223,979	716,843
4	558,217	229,169	713,934
5	543,291	237,459	713,592
Rata2	553,415	224,800	715,368

Pada tabel 6.4 dapat dilihat bahwa hasil pengujian rata-rata konsumsi daya pada masing-masing *library* terhadap gambar berukuran 3MB menunjukkan bahwa ketiga *library* memiliki hasil pengukuran konsumsi daya yang berbeda jauh antara satu *library* dengan *library* lainnya. Secara keseluruhan *library glide* tetap mencatatkan hasil pengukuran konsumsi daya paling rendah, bahkan cenderung konstan dan tidak terpengaruh dengan perbedaan pengujian dengan ukuran gambar yang lebih besar dengan rata-rata konsumsi daya sebesar 224,800 mW. Sementara pada pengujian kali ini, *library Picasso* dan *UIL* benar-benar terlihat terpengaruh dengan pengujian dengan ukuran gambar yang lebih besar, keduanya mencatatkan konsumsi daya yang meningkat cukup tajam dibanding pengujian dengan ukuran gambar yang lebih kecil. Secara keseluruhan hasil rata-rata pengukuran tetap menempatkan *library Picasso* sebagai *library* yang paling boros penggunaan dayanya dengan rata-rata konsumsi daya sebesar 715,368 mW dan menempatkan *library UIL* di posisi kedua rata-rata konsumsi daya sebesar 553,415mW.



Gambar 6.5 Komparasi Hasil Pengujian Konsumsi Daya 3 MB

6.3.4 Komparasi rata-rata hasil pengujian dengan berbagai ukuran gambar

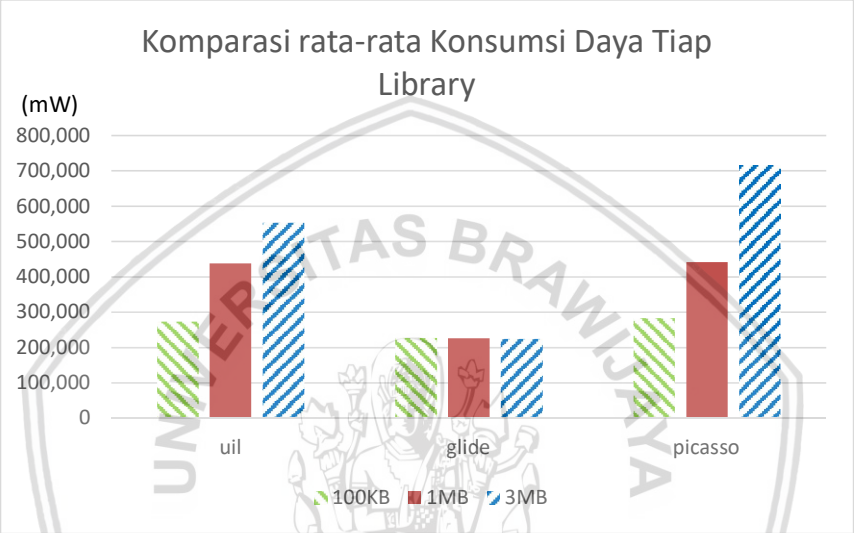
Tabel 6.5 Nilai rata-rata hasil pengujian dengan berbagai ukuran gambar (mW)

Ukuran	*uil*	*glide*	*picasso*
100KB	273,271	227,353	283,758
1MB	438,338	226,272	441,823
3MB	553,415	224,800	715,368
Rata2	421,675	226,142	480,316

Pada tabel 6.5 dapat dilihat bahwa perbandingan rata-rata hasil pengujian konsumsi daya *library Image loader* dengan berbagai ukuran gambar menghasilkan hal yang cukup berbeda antar *library*.

Secara keseluruhan dari ketiga *library* yang telah diuji, maka dapat diperoleh hasil bahwa *library* dengan konsumsi daya paling rendah adalah *library Glide* dengan rata-rata konsumsi daya sebesar 226,142mW. Dibandingkan dengan *library UIL* memiliki rata-rata konsumsi daya sebesar 421,601mW dan *library UIL* memiliki rata-rata konsumsi daya sebesar 480,316mW. Pada *library Picasso,* terjadi peningkatan rata-rata konsumsi daya sebesar 153,527mW ketika diuji dengan ukuran gambar sebesar 1MB, dan meningkat lagi sebesar 273,545mW ketika diuji dengan ukuran gambar sebesar 3MB. Sementara pada *library UIL,* terjadi peningkatan rata-rata konsumsi daya sebesar 165,287mW ketika diuji dengan ukuran gambar sebesar 1MB, dan meningkat lagi sebesar 115,076mW ketika diuji dengan ukuran gambar sebesar 3MB.

Sehingga dapat disimpulkan bahwa untuk *library glide* selalu konsisten sebagai *library* dengan konsumsi daya paling rendah dan bahkan mencatatkan hasil yang cukup konstan dan tak terpengaruh pengujian dengan berbagai ukuran gambar. Sementara *library UIL* dan *Picasso* terlihat terpengaruh dan mencatatkan nilai yang lebih tinggi jika diuji dengan gambar dengan ukuran yang lebih besar. Pada pengujian gambar dengan ukuran 100KB dan 1 MB, *library UIL* dan *Picasso* mencatatkan nilai yang tidak begitu jauh. Sementara pada pengujian dengan gambar dengan ukuran 3MB, *library UIL dan Picasso* mencatatkan hasil yang cukup berbeda jauh meskipun *library Picasso* tetap tercatat yang paling boros.



Gambar 6.6 Komparasi hasil rata-rata Konsumsi Daya Tiap *Library*

6.4 Analisis Hasil Pengujian

Analisis hasil pengujian merupakan sebuah tahap akhir dalam penelitian Analisis Perbandingan Konsumsi Daya *Library Image loader* pada *Android* (studi kasus: Aplikasi Media sosial). Analisis hasil pengujian bertujuan untuk menganalisis hasil dari semua pengujian. Hasil dari analisis adalah sebagai berikut:

1. *Library Glide* mendapatkan hasil pengujian konsumsi daya paling sedikit dalam semua pengujian dikarenakan mekanisme pada *library glide* yang melakukan proses kompresi gambar sehingga penggunaan *memory* menjadi lebih kecil (Yoo Jeong, et al, 2016). Mekanisme kompresi gambar yang tinggi dari *library Glide* juga membuat *library* ini hanya membutuhkan kineja *CPU* yang rendah yang berpengaruh pada konsumsi daya yang rendah. Kekurangan *library* ini resolusi lebih kecil sehingga detail gambar berkurang akibat dari kompresi gambar. Dengan kekurangan dan kelebihan seperti itu, *library Glide* cocok digunakan untuk aplikasi yang membutuhkan respon cepat dan tidak mementingkan detail ukuran gambar seperti aplikasi perpesanan/chatting.

2. *Library UIL* mendapat hasil pengujian konsumsi daya yang lebih besar daripada *library Glide* namun lebih kecil daripada *library Picasso*. Hal itu dikarenakan *library UIL* memiliki mekanisme kompresi gambar dan penggunaan *memory* yang lebih kecil daripada *library Picasso*. Hasil pengukuran kinerja *CPU* juga menunjukkan kalau *library UIL* mempunyai tingkat kinerja *CPU* yang lebih tinggi dari *library Glide* namun lebih rendah dari *library*

Picasso. Kekurangan *library* ini resolusi lebih kecil sehingga detail gambar berkurang akibat dari kompresi gambar. Sehingga *library UIL* ini paling cocok diterapkan pada aplikasi yang sedang dikembangkan karena banyak kustomisasi yang bisa dilakukan dan penggunaan *memory* yang sangat rendah (Hackernoon, 2015) dan cocok untuk aplikasi yang ringan namun butuh detail gambar yang cukup baik seperti aplikasi media sosial.

3. Library Picasso memperoleh hasil pengujian paling tinggi tingkat konsumsi dayanya dibandingkan kedua *library* lainnya karena *library Picasso* tidak memiliki mekanisme kompresi gambar sehingga mengunduh gambar langsung pada ukuran aslinya. *Picasso* membutuhkan *resource memory* yang lebih dan memungkinkan Out of *memory* terjadi, karena besarnya penggunaan *memory* oleh *bitmap* pada resolusi yang besar (Linares-Vásquez, 2015). Mekanisme pengunduhan gambar oleh *Picasso* ini juga mengakibatkan *library* ini membuat penggunaan *CPU* lebih besar karena semakin besar ukuran gambar maka semakin besar pula proses yang dilakukan oleh *CPU* yang akan membuat *library* ini mencatatkan konsumsi daya yang paling tinggi diantara 2 *library* lainnya. *Library Picasso* menampilkan gambar yang sama dengan yang dikirim oleh *server* karena tidak ada mekanisme kompresi sehingga dapat menampilkan gambar dengan detail yang tinggi. Dengan karakteristik seperti ini, *library Picasso* tidak cocok diimplementasikan pada aplikasi yang memerlukan banyak gambar dengan resolusi tinggi. *Library* ini Lebih cocok untuk aplikasi yang membutuhkan detail gambar yang tinggi dan sedikit gambar pada setiap *layout* halamannya seperti aplikasi galeri foto untuk menampilkan hasil foto setelah di abadikan oleh kamera.

4. Secara Keseluruhan, dapat diambil kesimpulan bahwa bebrapa faktor yang mempengaruhi tingkat konsumsi daya yang tinggi adalah tingginya tingkat penggunaan *memory* dan tingginya tingkat penggunaan *CPU*.

BAB 7 PENUTUP

Bagian ini memuat kesimpulan dan saran terhadap penelitian ini. Kesimpulan dan saran disajikan secara terpisah, dengan penjelasan sebagai berikut:

7.1 Kesimpulan

Dari hasil pengujian serta analisis yang telah dibuat sebelumnya, maka penelitian ini mendapatkan hasil sebagai berikut:

1. Komparasi penggunaan daya antar *library* dilakukan dengan cara membuat sebuah purwarupa aplikasi media sosial yang nantinya akan digunakan sebagai sistem *dummy* dalam pengujian penggunaan daya pada *library image loader* . Pengujian dilakukan selama 15 menit dan sebanyak 5 kali pada setiap *library*. Dilakukan juga pengujian yang sama pada setiap ukuran gambar yang telah ditentukan sebelumnya yaitu 100Kb, 1Mb, dan 3Mb. Dari pengujian tersebut akan diambil nilai rata-rata untuk mendapatkan hasil akhir yang digunakan sebagai nilai pembanding antar *library*.

2. Dari ketiga *library* yang telah diuji, maka dapat diperoleh hasil bahwa *library* dengan konsumsi daya paling rendah adalah *library Glide* dengan rata-rata konsumsi daya sebesar 225,780mW. Dibandingkan dengan *library UIL* memiliki rata-rata konsumsi daya sebesar 421,601mW dan *library Picasso* memiliki rata-rata konsumsi daya sebesar 481,829mW. Hal itu bisa terjadi karena pada *library Glide*, secara default gambar yang dimuat telah dikompres dengan baik sehingga tidak memberatkan kinerja *memory* dan *CPU*. Karena mekanisme tersebutlah akan berdampak terhadap konsumsi daya yang lebih rendah. Menurut hasil pengujian maka *library Glide* cocok diterapkan pada aplikasi perpesanan / chatting, sementara *library UIL* cocok diterapkan untuk aplikasi media sosial yang berfokus pada gambar seperti *instagram*, dan yang terakhir *library Picasso* cocok diterapkan pada aplikasi yang butuh detail gambar yang tinggi seperti aplikasi Galery foto.

3. Pengaruh perbedaan ukuran gambar pada *library Glide* tidak begitu terlihat karena nilai konsumsi dayanya pada setiap ukuran gambar tidak memiliki selisih yang banyak. Pengaruh perbedaan ukuran gambar yang diuji hanya berpengaruh pada tingkat konsumsi daya dari *library Picasso* dan *UIL*. Semakin besar ukuran gambar yang diuji semakin besar konsumsi daya yang dibutuhkan. Pada *library Picasso*, terjadi peningkatan rata-rata konsumsi daya sebesar 153,527mW ketika diuji dengan ukuran gambar sebesar 1MB, dan meningkat lagi sebesar 273,545mW ketika diuji dengan ukuran gambar sebesar 3MB. Sementara pada *library UIL*, terjadi peningkatan rata-rata konsumsi daya sebesar 165,287mW ketika diuji dengan ukuran gambar sebesar 1MB, dan meningkat lagi sebesar 115,076mW ketika diuji dengan ukuran gambar sebesar 3MB. Peningkatan yang terjadi pada *library Picasso* dikarenakan ukuran gambar yang diunduh sama persis dengan ukuran aslinya tanpa adanya mekanisme kompresi sehingga membuat kinerja *memory* dan *CPU* lebih berat dan mengakibatkan konsumsi daya berlebih,

sementara pada *library UIL* memiliki kompresi gambar namun tidak semaksimal *glide* sehingga membutuhkan *resource* yang lebih banyak.

7.2 Saran

Penelitian ini dapat dikembangkan dengan lebih baik dengan saran sebagai berikut:

1. Melakukan pengujian dengan menggunakan jaringan *public* sehingga dapat mengetahui tingkat konsumsi daya *library* jika menggunakan jaringan internet.
2. Menambahkan parameter pengujian performa seperti penggunaan *network* dan penggunaan *disk* (penyimpanan).
3. Melakukan pengujian dengan membandingkan penggunaan aplikasi yang menggunakan *library image loader* dengan aplikasi yang tidak menggunakan *library image loader*.

DAFTAR PUSTAKA

Android Developer (2017). Create an *Android Library* [online] <https://*developer*.*android*.com/studio/projects/*android-library*.html> [diakses 9 Maret 2017]

AppBrain. (2017). *Android image loader* s. [online] Tersedia di: <www.appbrain.com/stats/libraries/dev> [diakses 9 Maret 2017].

APJII. (2016). Infografis Penetrasi & Perilaku Pengguna Internet Indonesia. (Halaman: 23).

Azisubekti. (2016). Medsos Menggeser Media Konvensional? [online] <https://azissubekti.com/2016/08/20/medsos-menggeser-media-konvensional/> [diakses 9 Maret 2017]

Bakker, A. (2014). Comparing Energy Profilers for *Android*Mobile Devices. University of Twente.

Carlos, J., et al. (2013). Optimizing Energy Consumption per Application in Mobile Devices. IEEE.

Hackernoon. (2015). *Picasso*, *Universal image loader* or *Glide*? That's the question. [online] < https://hackernoon.com/*picasso*-universal-image-loader -or-*glide*-that-s-the-question-af34fa7f5e63 > [diakses maret 2018]

Glide Github. (-). *Glide*-readme.md. [online] Tersedia di: <https://github.com/bumptech/*glide*> [diakses 10 Maret 2017]

IN the Cheese Factory. (2015). Introduction to *Glide*, *Image loader Library* for *Android*, recommended by Google. [online] Tersedia di: <https://inthecheesefactory.com/blog/get-to-know-*glide*-recommended-by-google/en> [diakses 10 Maret 2017]

Kominfo. (2016). Kominfo: Pengguna Internet Indonesi 63 Juta Orang [online] <https://kominfo.go.id/index.php/content/detail/3415/kominfp+%3A+Pengguna+Internet_di+indonesia+63+juta+orang/0/berita_satker/> [diakses 9 Maret 2017]

Linares-Vásquez, et al. (2015). How *Developer*s Detect and Fix Performance Bottlenecks in *Android* Apps. IEEE.

Medium. (2017). Testing the battery Drain for *Android* App [online] https://medium.com/@pcloudy/testing-the-battery-drain-for-*android*-app-4d35516cd61a [diakses 4 Maret 2018]

Liutova, O. (2016). *Android* Mobile Application for Analysis of Cosmetic Products Composition. Czech Technical University in Prague.

Picasso Github. (2013). *Picasso*, a powerful image downloading and caching *library* for *Android*. [online] Tersedia di: < http://square.github.io/*picasso*/ > [diakses 10 Maret 2017]

Qualcomm *Developer*. (2013). *Trepn*Power Profiler <https://*developer*.qualcomm.com/software/trepn-power-profiler > [diakses 10 Januari 2018]

Raster Scracth-Wiki. (-). Raster Graphics. [online] Tersedia di: < https://en.scratch-wiki.info/wiki/Raster_Graphics > [diakses 12 Maret 2017]

Rong Wu,D., Yi Chen,K. (2014). The analysis of wireless throughput and power consumption on *Android* sistems. IEEE.

Sarungpreneur. (2015). Inilah Macam-macam Media sosial yang Populer di Dunia. [online] Tersedia di: < http://sarungpreneur.com/inilah-macam-macam-sosial-media-yang-populer-di-dunia/ > [diakses 9 Maret 2017]

Media sosial Today. (2017). How Much Time Do People Spend on Media sosial (Infographic). [online] Tersedia di: < https://www.sosialmediatoday.com/marketing/how-much-time-do-people-spend-sosial-media-infographic/ > [diakses 15 Maret 2017]

Sitepoint. (2013). Getting Started with *Android Library* Projects, Part 1. [online] Tersedia di: < https://www.sitepoint.com/getting-started-with-*android-library*-projects-part-1/ > [diakses 10 Maret 2017]

Stacktips. (2014). *Universal image loader Library* in *Android*. [online] Tersedia di: <http://stacktips.com/tutorials/*android*/universal-image-loader -*library*-in-*android*> [diakses 9 Maret 2017]

Song,Y.J., et al. (2015). A Study on Comparison Analysis of Performance and Usage between *Picasso* and *Glide*. Advanced Science and Technology Letters.

Trestian, R., et al. (2012). Energy Consumption Analysis of Video Streaming to *Android* Mobile Devices. IEEE.

Tutplus. (2014). *Android* SDK: Working with *Picasso*. [online] Tersedia di: < https://code.tutsplus.com/tutorials/*android*-sdk-working-with-*picasso*--cms-22149/ > [diakses 10 Maret 2017]

UIL Github. (2017). *Universal image loader* -readme.md. [online] Tersedia di: < https://github.com/nostra13/*Android*-Universal-Image-Loader > [diakses 9 Maret 2017]

Willockx, et al. (2016). Comparing performance parameters of mobile app

development strategies. IEEE.

Yang, P., Zhou,D. (2015). A Light Weight Energy Profiling Approach on *Android* Devices. ICMMCCE.

Yoo Jeong, et al. (2016). An Analysis of Existing *Android* Image Loading Libraries:

Picasso, *Glide*, Fresco, A*UIL* and Volley. IMEIA.